

Pascal 3.0 Source Code Listings

Volume II-Pascal

for the HP 9000 Series 200 Computers

Manual Part No. 98615-90074

© Copyright 1985, Hewlett-Packard Company.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company. The information contained in this document is subject to change without notice.

Use of this manual and flexible disc(s) or tape cartridge(s) supplied for this pack is restricted to this product only. Additional copies of the programs can be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

Restricted Rights Legend

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b)(3)(B) of the Rights in Technical Data and Software clause in DAR 7-104.9(a).

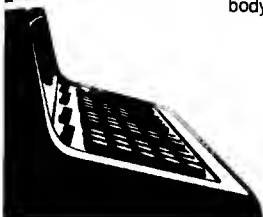
© Copyright 1980, Bell Telephone Laboratories, Inc.

© Copyright 1979, 1980, The Regents of the University of California.

This software and documentation is based in part on the Fourth Berkeley Software Distribution under license from the Regents of the University of California.

© Copyright 1979, The Regents of the University of Colorado, a body corporate.

This document has been reproduced and modified with the permission of the Regents of the University of Colorado, a body corporate.



Hewlett-Packard Company
3404 East Harmony Road, Fort Collins, Colorado 80525

Printing History

New editions of this manual will incorporate all material updated since the previous edition. Update packages may be issued between editions and contain replacement and additional pages to be merged into the manual by the user. Each updated page will be indicated by a revision date at the bottom of the page. A vertical bar in the margin indicates the changes on each page. Note that pages which are rearranged due to changes on a previous page are not considered revised.

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The manual part number changes when extensive technical changes are incorporated.

February 1985...Edition 1

Warranty Statement

Hewlett-Packard products are warranted against defects in materials and workmanship. For Hewlett-Packard computer system products sold in the U.S.A. and Canada, this warranty applies for ninety (90) days from the date of shipment. * Hewlett-Packard will, at its option, repair or replace equipment which proves to be defective during the warranty period. This warranty includes labor, parts, and surface travel costs, if any. Equipment returned to Hewlett-Packard for repair must be shipped freight prepaid. Repairs necessitated by misuse of the equipment, or by hardware, software, or interfacing not provided by Hewlett-Packard are not covered by this warranty.

HP warrants that its software and firmware designated by HP for use with a CPU will execute its programming instructions when properly installed on that CPU. HP does not warrant that the operation of the CPU, software, or firmware will be uninterrupted or error free.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. HEWLETT-PACKARD SHALL NOT BE LIABLE FOR CONSEQUENTIAL DAMAGES.

HP 9000 Series 200

For the HP 9000 Series 200 family, the following special requirements apply. The Model 216 computer comes with a 90-day, Return-to-HP warranty during which time HP will repair your Model 216, however, the computer must be shipped to an HP Repair Center.

All other Series 200 computers come with a 90-Day On-Site warranty during which time HP will travel to your site and repair any defects. The following minimum configuration of equipment is necessary to run the appropriate HP diagnostic programs: 1) 1/2 Mbyte RAM; 2) HP-compatible 3 1/2" or 5 1/4" disc drive for loading system functional tests, or a system install device for HP-UX installations; 3) system console consisting of a keyboard and video display to allow interaction with the CPU and to report the results of the diagnostics.

To order or to obtain additional information on HP support services and service contracts, call the HP Support Services Tele-marketing Center at (800) 835-4747 or your local HP Sales and Support office

* For other countries, contact your local Sales and Support Office to determine warranty terms.

Table of Contents

File→Module Map	1
Module→File Map	3
Cross Reference	7
A804XDVR	189
AMIGO	195
ASCII	207
BAT	211
BUBBLES	213
C_HOOK	217
CLOCK	221
CONVERT	225
CRT	229
CS80	237
CTABLE	253
DC_DRV	273
DGL_C_IN	283
DGL_C_OUT	287
DGL_HPGL	293
DGL_INQ	301
DGL_KNOB	309
DGL_POLY	313
DGL_RAS	325
DGL_TOOLS	333
DGL_VARS	335
DI_DRV	341
DISCHPIB	347
DMA_DRV	355
EDRIVER	363
EPROMS	367
ETU	371
F9885	383
GCRT	389
G_DRV	395
GEN	401
GLE_FILE	409
GLE_GEN	413
GLE_GENI	417
GLE_HPGL	421
GLE_HPGLI	431
GLE_HPIB	437
GLE_KNOB	445
GLE_RGL	449
GLE_SCLIP	455
GLE_SMARK	457
GLE_STEXT	461
GLE_TYPES	465

GLE_UTLS	473
H_DRV	477
HEAPT	485
HPHIL	489
INIT	495
INITLOAD	507
IOLIB	525
KERNEL	555
KEYS	573
LIB	581
LIBRARIAN	595
LIFDAM	629
LOCKMOD	641
M68KSYS	643
MINIT	655
MIUI	669
MOUSE	677
NONUSKBD1	681
NONUSKBD2	685
PRINTER	689
RS_DRV	695
SEGMENTER	701
SRMAM	705
SRMDAM	709
SRM_DRV	727
SWVOL	747
SYSDEVS	749
TAIL	757
TAPEBKUP	759
TYPES	777

File → Module Map

The title pages in the volumes of listings refer to *file* names, not *volume* names. The compilation of a single file may include multiple modules, and the module names many times bear no resemblance to the file names in which they appear. To help you determine which modules appear in each file of this listings volume, the following file:module cross-reference has been created. In the left column are the file names, and in the right column, in alphabetical order, are the modules which are declared in each file. The map only includes files whose listings are in this volume of listings.

A804XDVR	a804xdvr
AMIGO	amigodvr, csamigo
ASCII	asciimodule
BAT	bat
BUBBLES	bubble
CLOCK	clock
CONVERT	convert_text
CRT	crt
CS80	cs80, cs80dsr, cs80dvr, tapebuf
CTABLE	brstuff, ctr, options, scanstuff
C_HOOK	(no explicit modules)
DC_DRV	extdc, init_dc, intdc
DGL_C_IN	dgl_config_in
DGL_C_OUT	dgl_config_out
DGL_HPGL	dgl_hpgl
DGL_HPGLI	dgl_hpgli
DGL_INQ	dgl_inq
DGL_KNOB	dgl_knob
DGL_POLY	dgl_poly
DGL_RAS	dgl_raster
DGL_TOOLS	dgl_tools
DGL_VARS	dgl_vars
DISCHPIB	bkgnd, dischpib
DI_DRV	extdi, init_discint
DMA_DRV	init_dma
EDRIVER	edriver
EPROMS	eproms
ETU	(no explicit modules)
F9885	f9885dvr
GCRT	crtb
GEN	dgl_gen
GLE_FILE	gle_file_io
GLE_GEN	gle_gen
GLE_GENI	gle_geni
GLE_HPGL	gle_hpgl_out
GLE_HPGLI	gle_hpgl_in
GLE_HPIB	gle_hpib_io
GLE_KNOB	gle_knob_in
GLE_RGL	gle_ras_out

GLE_SCLIP	gle_sclip
GLE_SMARK	gle_smark
GLE_STEXT	gle_stext
GLE_TYPES	gle_types
GLE_UTLS	gle_utls
G_DRV	extg, init_gpio
HEAPT	hpm, sysglobals
HPHIL	hphil
H_DRV	exth, init_hpib
INIT	fs, initunits, isr, ldr, misc
INITLOAD	bootdammodule, loader, mini, sysglobals
IOLIB	general_1, general_2, general_3, general_4, hpib_0, hpib_1, hpib_2, hpib_3, serial_0, serial_3
KERNEL	general_0, iocomasm,iodeclarations
KEYS	keys
LIB	dgl_lib
LIBRARIAN	(no explicit modules)
LIFDAM	lifmodule
LOCKMOD	lockmodule
M68KSYS	ci
MINIT	asmr, bminit, cs80ir, hminit, iramigo, midecs, mminit, qminit, xminit,
MIUI	(no explicit modules)
MOREFSYS	mfs
MOUSE	mouse
NONUSKBD1	non_us_kbd1
NONUSKBD2	(no explicit modules)
PRINTER	prtdvr
RS_DRV	init_rs, rs
SEGMENTER	asm, segmenter
SRMAM	srمامodule
SRMDAM	srmdammodule
SRM_DRV	srم
SWVOL	(no explicit modules)
SYSDEVS	sysdevs
TAIL	(no explicit modules)
TAPEBKUP	cs80tbdvr, cs80tbr

Module → File Map

The following cross-reference map is the inverse of the one in the preceding section. This map will tell you which listing file a particular module is in.

A804XDVR	
amigodvr	AMIGO
asciimodule	ASCII
asm	SEGMENTER
asmr	MINIT
bat	BAT
bkgnd	DISCHPIB
bminit	MINIT
bootdammodule	INITLOAD
brstuff	CTABLE
bubble	BUBBLES
ci	M68KSYS
clock	CLOCK
convert_text	CONVERT
crt	CRT
crtb	GCRT
cs80	CS80
cs80dsr	CS80
cs80dvr	CS80
cs80ir	MINIT
cs80tbdvr	TAPEBKUP
cs80tbr	TAPEBKUP
csamigo	AMIGO
ctr	CTABLE
dgl_config_in	DGL_C_IN
dgl_config_out	DGL_C_OUT
dgl_gen	GEN
dgl_hpgl	DGL_HPGL
dgl_hpgli	DGL_HPGLI
dgl_inq	DGL_INQ
dgl_knob	DGL_KNOB
dgl_lib	LIB
dgl_poly	DGL_POLY
dgl_raster	DGL_RAS
dgl_tools	DGL_TOOLS
dgl_vars	DGL_VARS
dischpiib	DISCHPIB
edriver	EDRIVER
eproms	EPROMS
extdc	DC_DRV
extdi	DI_DRV
extg	G_DRV
exth	H_DRV
f9885dvr	F9885

fs	INIT
general_0	KERNEL
general_1	IOLIB
general_2	IOLIB
general_3	IOLIB
general_4	IOLIB
gle_file_io	GLE_FILE
gle_gen	GLE_GEN
gle_geni	GLE_GENI
gle_hpgl_in	GLE_HPGLI
gle_hpgl_out	GLE_HPGL
gle_hpib_io	GLE_HPIB
gle_knob_in	GLE_KNOB
gle_ras_out	GLE_RGL
gle_sclip	GLE_SCLIP
gle_smark	GLE_SMARK
gle_stext	GLE_STEXT
gle_types	GLE_TYPES
gle_utls	GLE_UTLS
hminit	MINIT
hphil	HPHIL
hpib_0	IOLIB
hpib_1	IOLIB
hpib_2	IOLIB
hpib_3	IOLIB
hpm	HEAPTHEAPT
init_dc	DC_DRV
init_discint	DI_DRV
init_dma	DMA_DRV
init_gpio	G_DRV
init_hpib	H_DRV
init_rs	RS_DRV
initunits	INIT
intdc	DC_DRV
iocomasm	KERNEL
iodeclarations	KERNEL
iramigo	MINIT
isr	INIT
keys	KEYS
ldr	INIT
lifmodule	LIFDAM
loader	INITLOAD
lockmodule	LOCKMOD
mfs	MOREFSYS
midecs	MINIT
mini	INITLOAD
misc	INIT
mminit	MINIT
mouse	MOUSE
non_us_kbd1	NONUSKBD1

options	CTABLE
prtdvr	PRINTER
qminit	MINIT
rs	RS_DRV
scanstuff	CTABLE
segmenter	SEGMENTER
serial_0	IOLIB
serial_3	IOLIB
srm	SRM_DRV
srمامodule	SRMAM
srmdammodule	SRMDAM
sysglobals	HEAPT
sysglobals	INITLOAD
sysdevs	SYSDEVS
tapebuf	CS80
xminit	MINIT

active controller						
GLE_HPIB(gle_hpib_io)	10142d	10148c	10151c	10236c	10324c	
IOLIE(hplib_1)	397d	466d	472c	475c	558c	646c 715c 765c
IOLIE(hplib_2)	1824c	1875c				
IOLIE(hplib_3)	2066c	2122c				
active isc						
DC_DRV(intdc)		434c				
DLSCHPIS(dlschpib)	***	467c				
IOLIB(general_4)	1270c	1300c	1320c	1333c	1336c	1378c 1569c 1596c 1662c
KERNEL(iodeclarations)	***	572d				
SRM_DRV(srm)	1197c	1275c	1290c	1318c		
actual						
SRM_DRV(srm)	483d	721d	814d	1452c		
actual_num_files						
SRMDAM(srmdanmodule)	595c	597c				
SRM_DRV(srm)	***	385d				
actual_num_passwords						
SRMDAM(srmdanmodule)	534c	536c				
SRM_DRV(srm)	***	407d				
actual_lfr_type						
KERNEL(iodeclarations)	559d	573d				
actual_size						
LIFDAM(lifdmodule)	485d	507c	508c			
add						
INITLOAD	1727d	1764c	1765c	1771c		
LIBRARIAN	1486d	1491c	1494c	1503c	1505c	1522c
add_err_data						
GLE_HPGLI(gle_hpgl_out)	7065d	7105c	7110c	7111c	7126c	7178c 7238c 7309c 7313c 7315c 7317c 7319c 7340c 7344c
	7366c	7370c	7409c	7442c	7455c	7476c 7495c 7497c 7571c 7579c 7614c
GLE_HPGLI(gle_hpgl_in)	18038d	18071c	18104c	18144c	18204c	18207c 18209c 18211c 18213c 18251c 18261c 18283c 18297c 18307c
	18313c	18344c	18352c			
addr_parm_data						
GLE_HPGI(gle_hpgl_out)	7080d	7314c	7316c	7318c	7320c	7343c 7345c 7369c 7371c 7410c 7448c 7452c 7454c 7456c
	7496c	7498c				
GLE_HPGII(gle_hpgl_in)	18053d	18208c	18210c	18212c	18214c	
addit						
INITLOAD(loader)	***	865d				
LIBRARIAN		188c	1523c	1614c	2384c	2388c 2413c
addr1						
GLE_RGL(gle_ras_out)	***	8027d				
addr2						
GLE_RGL(gle_ras_out)	***	8028d				
addr3						
GLE_FGL(gle_ras_out)	***	8029d				
addr_record						
AMIGO(amigodvr)	690c	959c				
AMIGO(csamigo)	151d	392d				
addr_record_cmd						
AMIGO(csamigo)	51d	401c				
addr_record_cmd_buf						
AMIGO(csamigo)	394d	400c	401c			
addr_to_listen						
GLE_HPILB(gle_hpilb_io)	10275d	10357c	10516c			
IOLIB(general_2)	920c	997c	1029c	1042c		
IOLIB(general_4)	***	1415c				
IOLIB(hplib_1)	405d	597d	679c	752c		
addr_to_talk						
GLE_HPILB(gle_hpilb_io)	10188d	10269c	10537c			
IOLIB(general_2)	853c	941c	972c	1016c	1055c	1076c
IOLIB(general_4)	***	1401c				
IOLIB(hplib_1)	403d	510d	591c	703c		

[illegible]


```

a-phatype
  CRT(crt) *** 708c
  C_HOOK *** 142c
  DGL_C_OUT(dgl_cfg_out) 11286c 11296c 11335c 11342c 11346c
  KEYS(keys) 233d 239d
  SYSDEVS(sysdevs) *** 113d
a1readyopen
  SRMDAM(srmdammodule) 660d 708c 820c 823c 843c 849c
a1tmode
  CRT(crt) 102d 148d
  GCRT(crtb) *** 81d
  SYSDEVS(sysdevs) *** 96d
a1u
  KEYS(keys) 150d 169c
a1vinit
  DC_DRV(extdc) *** 199d
  DC_DRV(intdc) *** 307c
a1vinit
  DC_ORV(extdc) *** 198d
  DC_ORV(init_dc) *** 676c
am
  ETU 436c 458c 479c
  INIT(misc) 469c 470c 471c 472c 473c
  INITLOAD(bootdammodule) 713c 714c
  INITLOAD(loader) 1120c 1611c
  INITLOAD(sysglobals) *** 97d
  LIBRARIAN 133c 139c 2742c
  LIFORAM(lifmodule) 296c 317c 920c 922c 923c 567c
  M8KSYS(c1) 119c 334c 335c 547c 548c
  SRMDAM(srmdammodule) 948c 949c 950c 1503c
  UCSD_AM(ucsdmodule) 82c 155c 164c 165c 166c 500c 503c 504c 623c 676c
amigo_class
  CTABLE(scanstuff) 1096d 1121c 1125c
amigo_class_type
  CTABLE(scanstuff) 1086d 1096d
amigo_dev_type
  AMIGO(amigodvr) 493d 506d
  AMIGO(csamigo) 43d 137d 160d 205d
amigo_fm_name
  CTABLE(ctr) 364d 707c 716c 733c 752c
amigodvr
  AMIGO(amigodvr) *** 422d
amigoinit
  AMIGO *** 32d
amigodc
  AMIGO(amigodvr) 430d 554d

```

```

amrequesttype
  AMIGO(amigodvr) *** 430d
  ASCII(asciimodule) 8d 13d
  BUBBLES(bubble) 29d 53d
  CRT(crt) *** 402d
  CS80(cs80dvr) 987d 1094d 1168d 1214d
  EPROMS(eproms) 8d 12d
  F9885(f9885dvr) *** 40d
  GCRT(crtb) *** 272d
  GLE_KNOB(gle_knob_in) *** 18047d
  INIT(initunits) 2139d 2148d 2152d 2156d 2228d
  INIT(misc) 483d 515d 534d
  INITLOAD(bootdammodule) 544d 747d
  INITLOAD(mini) 309d 384d
  INITLOAD(sysglobals) 66d 68d
  KEYS(keys) *** 72d
  M8KSYS(c1) 306d 731d
  PRINTER(prtdvr) 40d 47d
  SRMDAM(srmdammodule) 40d 178d
  SYSDEVS(sysdevs) 372d 376d 407d 432d 484d
  UCSD_AM(ucsd_am) *** 40d
  UNITIO(uio) 56d 77d
amtable
  ASCII *** 319c
  INIT(misc) 471c 472c 473c 700c 706c 720c
  INITLOAD *** 1705c
  INITLOAD(bootdammodule) 713c 714c
  INITLOAD(sysglobals) *** 262d
  LIFORAM(lifmodule) 296c 920c 922c 923c
  SRMDAM(srmdammodule) 948c 949c 950c
  UCSD_AM(ucsd_am) *** 271c
  UCSD_AM(ucsdmodule) 82c 164c 165c 166c
amtableptrtype
  INITLOAD(sysglobals) 173d 262d
amtabletype
  INITLOAD(sysglobals) 168d 173d
amtype
  CTABLE(ctr) *** 453d
  INIT(initunits) *** 2183d
  INITLOAD(sysglobals) 68d 97d 145d 168d 279d
  SYSDEVS(sysdevs) 119d 154d
an_15rib
  DMA_DRV(init_dma) 169d 170d 182d 199d
answer
  ETU 191d 197c 202c 203c 205c 206c 208c 213d 215c 221d 245c 259c 260c 289d
  MINIT(midecs) 291c 869d 1009c 1010c
  TAPEBKUP 43d 47c 48c 49c 50c
  any_9895_unit_missing 974d 979c 980c 981c 982c 983c
any_ctable
  CTABLE 1339d 1350c 1403c
any_to_ucsd
  CONVERT(convert_text) 35d 40d
anychange
  LIFORAM(lifmodule) 199d 320c 353c 375c 415c 428c 457c 461c 475c 699c 709c 713c 1095c 1124c
  1145c 1170c
anychar
  GLE_TYPES(gle_types) 1009d 1010d
  KEYS(keys) 52d 351c 352c 500c 536c 583c
anychar_ptr
  GLE_HPGL(gle_hpgl_out) 7067d 7141d
  GLE_HPGL(gle_hpgl_in) 18040d 18080d
  GLE_TYPES(gle_types) *** 1010d
  GLE_UTLS(gle_...) 21009d 21010d 21011d 21012d 21039d 21069d 21106d 21118d 21119d

```


archsavehook										
KEYS(keys)	55d	352c	501c							
anytomen										
ETU	418d	1043c								
aoff										
LIBRARIAN	1807d	1821c	1824c	1831c						
area_draw_mode										
GLE_RGL(gle_ras_out)	8049d	8514c								
areaisdbcrf										
CRT(crt)	635c	647c	654c							
GCRF(crb)	474c	485c								
GYSDEVS(sysdevs)	***	62d								
arevoalivpack										
SRRMAM(srmammodule)	***	155c								
SRM_DRV(srm)		1004d	1529d							
arithop										
LIBRARIAN	561d	1075c								
arithoptype										
LIBRARIAN	549d	561d								
arp										
INIT(ldr)	***	2376c								
INITLOAD(loader)		917d	1164c	1393c	1603c					
LIBRARIAN	***	2965c								
ascii_buffer										
DGL_C_IN(dgl_cfg_in)	***	12022d								
DGL_C_OUT(dgl_cfg_out)	***	11034d								
GLE_FILE(gle_file_io)		9029d	9031d							
GLE_HPGL(gle_hppl_out)		7013d	7015d							
GLE_HPGLI(gle_hppl_in)		18013d	18015d							
GLE_HPGLB(gle_hpbl_io)		10040d	10042d							
ascii_buffer_ptr										
DGL_C_IN(dgl_cfg_in)		12071d	12085d							
DGL_C_OUT(dgl_cfg_out)		11372d	11404d							
GLE_FILE(gle_file_io)		9029d	9065c							
GLE_HPGL(gle_hppl_out)		7013d	7044c	7060c	7072c	7086c	7122c	7144c	7171c	7228c
		7472c	7491c	7519c						
GLE_HPGLI(gle_hppl_in)		18013d	18032c	18045c	18059c	18083c	18102c	18139c	18249c	18295c
GLE_HPGLB(gle_hpbl_io)		10040d	10514c	10535c	18332c					
ascii_bufferspace										
DGL_C_IN(dgl_cfg_in)		12022d	12099c							
DGL_C_OUT(dgl_cfg_out)		11034d	11424c							
ascilam										
ASCII	***	319c								
ASCII(asciimodule)		6d	13d							
asciifile										
ASCII	***	319c	320c	321c						
ETU	***	930c								
INITLOAD(sysglobals)	***	50d								
asciimodule										
ASCII	***	317d								
ASCII(asciimodule)	***	3d								

[illegible]

[illegible]

csdf1.e													
INIT(misc)		714c	715c										
INITLOAD(sysglobals)	***	47d											
LIFUAM(lifmodule)	***	421c											
UCSD_BAM(ucsdmodule)	***	595c											
badio													
EDRIVER(edriver)		105d	114c	116c	131c	135c	138c	141c	190c	220c	235c	249c	276c
ETU		119d	125c	133c	163c	168c	203c	329c	552c	1004c			291c
LIFUAM(lifmodule)		80d	103c	171c	177c	189c	222c	309c	394c	488c	508c	509c	882c
		1068c	1071c	1073c	1086c	1162c						896c	1065c
badmessage													
ETU		123d	649c	662c	780c	783c	803c	921c	991c				
badverify													
F988S(f9885dvr)	***	48d											
baseaddr													
ETU		48d	672c	675c	689c	1185c	1197c	1211c	1219c				
basicchintype													
SRUAM(srmdamodule)		67d	939c										
basicproctype													
SRUAM(srmdamodule)		68d	940c										
bat													
BAT	***	99d											
BAT(bat)	***	32d											
bat8041cmdreg													
BAT(bat)		48d	76c										
bat8041datareg													
BAT(bat)		48d	70c	87c									
bat8041statusreg													
BAT(bat)		47d	55c	63c									
batbyterecv													
CLOCK(clock)		148c	149c	153c	154c	155c							
SYSDEVS(sysdevs)		381d	440d	442c									
batcmdhook													
BAT(bat)	***	93c											
SYSDEVS(sysdevs)		190d	438c	607c									
batcmdtype													
SYSDEVS(sysdevs)		185d	190d										
batcommand													
BAT(bat)	***	92c											
CLOCK(clock)		96c	97c	147c	148c	149c	153c	154c	155c				
SYSDEVS(sysdevs)		380d	437d										
batdataout													
BAT(bat)		69d	77c	78c	79c	80c	81c						
batinit													
BAT	***	103c											
BAT(bat)		35d	90d										
batreadbyte													
BAT(bat)		85d	94c										
batreadhook													
BAT(bat)	***	94c											
SYSDEVS(sysdevs)		191d	442c	606c									
batreadtype													
SYSDEVS(sysdevs)		187d	191d										
batstatus													
BAT(bat)		52d	55c	56c	60d	63c	64c						
batterypresent													
BAT	***	102c											
BAT(bat)	***	73c											
CLOCK(clock)	***	141c											
SYSDEVS(sysdevs)	***	189d											
bba													
CS80(cs80)	***	222d											
TAPEBKUP(cs80tdvr)		637c	639c										
bbadcount													
BUBBLES(bubble)	***	45c											
bbaddata													
BUBBLES(bubble)	***	47c											
bbadinterrupt													
BUBBLES(bubble)	***	43c											
bbadsrctor													
BUBBLES(bubble)	***	44c											
bbuffer													
BUBBLES(bubble)		76c	93c										
bcb													
DC_DRV(intdc)		254d	408d	416c									
DI_DRV(extdi)	***	165d											
G_DRV(extg)	***	182d											
H_DRV(exth)	***	180d											
RS_DRV(rs)	***	182d											
bcd													
CS80(cs80dsr)		781d	919c										
CS80(cs80dvr)		1034d	1064c										
INITLOAD(bootdamodule)		555d	558d										
LIFUAM(lifmodule)		37d	41d										
MINIT(qminit)		941d	1008c										
MIU		389d	390d										
MORF SYS(mfs)	***	552d											
TAPEBKUP(cs80tdvr)		460d	486c										
bcd12													
MIU		390d	404d	413d	426d								
bcd_prod_num													
CS80(cs80dsr)		778d	916c	919c									
CS80(cs80dvr)		1031d	1061c	1084c									
bcd_product_number													
MINIT(qminit)		938d	1005c	1008c									
TAPEBKUP(cs80tdvr)		457d	483c	486c									
bcd_srt													
MORF SYS(mfs)		399d	406c	503c	562d	582c	583c	605c					
bcd_srttype													
MORF SYS(mfs)		32d	40d	399d	552d	556d	562d						
bcdigit													
MORF SYS(mfs)		31d	34d										
bcount													
BUBBLES(bubble)		78c	80c	96c	97c	99c							
bdatype													
SRUAM(srmdamodule)		66d	938c	1047c									
bduration													
A804XDVR(a804xdvr)	***	388c											
SYSDEVS(sysdevs)		173d	414c										
bdx_chain_semaphore													
DISC(HPIB(bkgnd)	***	52d											
DISC(HPIB(dischpio)		486c	498c	501c									
bdx_nl													
DISC(HPIB(bkgnd)	***	54d											
DISC(HPIB(dischpio)		461c	513c	514c	516c	517c	529c	534c					
bdx_pre_eoi													
ARIG0(amigodvr)	***	942c											
CS80(cs80dvr)		1122c	1128c										
DISC(HPIB(bkgnd)	***	53d											
DISC(HPIB(dischpio)		530c	531c	533c	534c								
bdx_proc													
DISC(HPIB(bkgnd)	***	55d											
DISC(HPIB(dischpio)		462c	542c										

bincmp												
IOLIB(serial_0)												
KERNEL(iocomasm)	***	2317c	2338c									
binear												
KERNEL(iocomasm)	***	709d										
binior												
IOLIB(serial_0)		2254c	2275c									
KERNEL(iocomasm)	***	707d										
bint												
ETU		840d	954c	1170d	1185c	1190c	1191c	1197c	1211c	1215c	1216c	1219c
biofail												
BUBBLES(bubble)	***	48c										
bip												
DISCHPIB(bkgnd)		94d	100c	107d	111c	114c	115c	118c				
bip1												
ARI_type		598c	675c	702c	720c	766c	777c	799c	938c			
CS80(cs80dvr)		1090c	1101c									
DISCHPIB(bkgnd)		58d	94d									
DISCHPIB(dischnpb)		447c	497c	508c	526c	130c	149c	157c	158c	165c		
MINIT(cs80ir)		759c	806c									
MINIT(hminit)	***	308c										
MINIT(qminit)		984c	1133c									
TAPEBKUP(cs80tbdvr)	***	735c										
TAPEBKUP(cs80tbr)	***	284c										
bip_valld												
DISCHPIB(bkgnd)		94d	98c	101c	129c	157c	165c					
bit												
CS80(cs80dsr)		786d	936c									
MINIT(qminit)		947d	1017c									
TAPEBKUP(cs80tbdvr)		466d	493c									
bit0												
C_HOOK	***	46d										
DGL_TOOLS(dgl_tools)	***	20024d										
GLE_HPGLI(gle_hpgl_in)	***	18243d										
GLE_RGL(gle_ras_out)	***	8264c										
GLE_TYPES(gle_types)	***	1246d										
bit1												
C_HOOK	***	46d										
DGL_TOOLS(dgl_tools)	***	20024d										
GLE_HPGLI(gle_hpgl_in)	***	18242d										
GLE_TYPES(gle_types)	***	1246d										
bit10												
GLE_TYPES(gle_types)	***	1245d										
bit11												
GLE_HPGLI(gle_hpgl_in)	***	18232d										
GLE_TYPES(gle_types)	***	1245d										
bit12												
GLE_HPGLI(gle_hpgl_in)	***	18232d										
GLE_TYPES(gle_types)	***	1245d										
bit13												
C_HOOK	***	44d										
DGL_TOOLS(dgl_tools)	***	20022d										
GLE_HPGLI(gle_hpgl_in)	***	18232d										
GLE_TYPES(gle_types)	***	1245d										
bit14												
C_HOOK	***	44d										
DGL_TOOLS(dgl_tools)	***	20022d										
GLE_HPGLI(gle_hpgl_in)	***	18232d										
GLE_TYPES(gle_types)	***	1245d										
bit15												
GLE_HPGLI(gle_hpgl_in)	***	18232d										
GLE_TYPES(gle_types)	***	1245d										


```

bit2
C_HOOK *** 46d
DGL_TOOLS(dgl_tools) *** 20024d
bit3
C_HOOK *** 46d
DGL_TOOLS(dgl_tools) *** 20024d
bit4
C_HOOK *** 46d
DGL_TOOLS(dgl_tools) *** 20024d
GLE_TYPES(gle_types) *** 1246d
bit5
C_HOOK *** 46d
DGL_TOOLS(dgl_tools) *** 20024d
bit6
C_HOOK 39d 46d
DGL_TOOLS(dgl_tools) 20017d 20024d
GLE_TYPES(gle_types) *** 1246d
bit7
C_HOOK 39d 46d
DGL_TOOLS(dgl_tools) 20017d 20024d
bit8
GLE_TYPES(gle_types) *** 1245d
LIBRARIAN 438d 658c 849c 1006c 1011c 1048c 1081c 1096c 1101c 1104c 1113c
bit9
GLE_TYPES(gle_types) *** 1245d
bit_mask
C_HOOK 81d 101c 104c 105c
DGL_RAS(dgl_raster) 17382d 17409c 17412c 17413c
GCRT(crtb) 154d 177c 180c 181c
bit_set
GLE_HPIB(gle_hpib_10) 10148c 10243c 10252c 10259c 10261c 10331c 10340c 10347c 10349c
IOIB(hpib_1) 472c 487c 565c 574c S81c 583c 653c 662c 669c 671c
IOIB(hpib_3) 2138c 2148c 2157c 2167c
IOIB(serial_0) 2412c 2460c
KERNEL(iocomasm) *** 703d
bitcount
LIBRARIAN 752d 762c 773c 785c 791c 800c
bitmap
INITLOAD(loader) 911d 995d
bitmapaddr
GCRT(crtb) 536c 543c
SYSEVS(sysdevs) *** 131d
bitmaptype
DGL_CONFIG(dgl_config_out) 11183c 11288c 11302c 11332c
GCRT(crtb) *** 520c
SYSEVS(sysdevs) *** 113d
bitop
LIBRARIAN 653d 673c 678c
bitoptype
LIBRARIAN 651d 653d

```

```

bkand
AMIGO(amigodvr) *** 425d
AMIGO(csamigo) *** 37d
CS80(cs80) *** 66d
CS80(cs80dvr) *** 598d
CS80(cs80dvr) *** 972d
CS80(tapebuf) *** 35d
DISCHPIB *** 553d
DISCHPIB(bkand) *** 28d
DISCHPIB(dischpib) *** 189d
MINIT(cs80ir) *** 516d
MINIT(hminit) *** 248d
MINIT(iramigo) *** 187d
MINIT(qminit) *** 878d
MIUI *** 11d
TAPEBKUP *** 953d
TAPEBKUP(cs80tdvr) *** 305d
TAPEBKUP(cs80tbr) *** 14d
bkand_and_dischpib_present
CTABLE(scanstuff) 1080d 1112c 1165c
blank
MOREFSYS(mfs) 51d 96c 370c 391c
blankchk 868d 954c 955c 956c
ETU
blanks
ETU 1163d 1178c 1184c 1186c 1193c 1198c 1212c 1218c
blink1_reg
GLE_TYPES(gle_types) *** 1275d
blink2_reg
GLE_TYPES(gle_types) *** 1276d
block
CS80(cs80dvr) 1168d 1172c 1177c 1220d 1244c 1250c 1267c 1271c 1278c 1282c 1293c 1298c 1307c 1310c
INIT(misc) 537d 556c S59c S92c 627c 628c 652c 663c 669c
INITLOAD(loader) 1118d 1120c
LIBRARIAN 131d 133c 137d 139c
UCSD_AM(ucsd_am) 71d 75c 76c 82c 94c 135d 142c 144c
UNITIO(ucio) 29d 31d 54d 65c 75d 86c
block_address
CS80(cs80dvr) 1095d 1110c
block_boundaries
CTABLE *** 1570c
CTABLE(ctr) 315d 621d 623c 624c
block_os
CTABLE(ctr) 330d 331d 686d 692c 698d 705c
blockpower
CS80(cs80dvr) 1218d 1236c 1237c 1240c 1244c 1245c 1247c 1271c 1298c 1302c 1310c
blocks
LIBRARIAN 131d 133c
UCSD_DAM(ucsdmodule) 186d 192c 194c 195c 217c 218c 235c 570d 582c 583c 584c 603c
blocksize
CS80(cs80dvr) 1219d 1240c 1241c 1257c 1275c
INITLOAD(loader) 813d 1142c 1145c 1147c 1148c 1149c 1152c 1153c 1154c 1322c 1325c 1326c 1328c 1330c
LIBRARIAN 1366d 1980c 1984c 2116c 2174c 2206c 2211c 2215c 2230c 2231c 2253c 2259c 2269c 2270c
2288c 2289c 2290c 2293c 2301c 2514c 2522c 2524c 2849c 2850c 2867c 2882c 3152c 3189c
3190c
M6KSYS(ci) *** 806c
blocktime
CS80(cs80) *** 104d
blockwrite
LIBRARIAN 131d 1985c 2118c 2207c 2212c 2232c 2271c 2868c 2883c 3112c
blu_intensity
GFL_RGL(gle_ras_out) 8045d 8199c

```


bu write_cmd																
AMIGO(amigodvr)	**	726c														
AMIGO(csamigo)	***	59d														
bufblocks																
LIBRARIAN		2834d	2849c	2850c	2865c	2866c										
bufend																
ASCII(asciimodule)		16d	21c	22c	37c	50c	59c	70c	72c	74c	85c	93c	119c	158c	202c	
		222c	247c	248c												
buffer																
A804XDVR(a804xdvr)		126d	128c	151d	154c											
AMIGO(amigodvr)		431d	625c													
ASCII(asciimodule)		7d	14d	185c	187c	194c	203c	223c	225c	251c	270c					
BUBBLES(bubble)		30d	54d	76c	93c											
CRT(crt)		402d	409c	415c	420c	430c										
CS80(cs80dvr)		988d	1395c													
EDRIVER(edriver)		38d	76d	199c	215c	217c	220c	230c	232c	235c	242c	243c	285c	296c	308d	
		322c														
EPROMS(eproms)		9d	13d	61c	68c											
ETU		419d	427c	564d	572c	578c	593c	603d	609c							
F9885(f9885dvr)		40d	110d	281c												
GCRT(crtb)		272d	279c	285c	290c	300c										
INIT(initunits)		2139d	2148d	2152d	2156d	2228d										
INIT(misc)		484d	503c	505c	516d	521c	525c	526c	535d	600c	605c	629c	631c	646c	649c	
INITLOAD(bootdammodule)		659c	664c	675c	677c	684c										
INITLOAD(mini)		545d	592d	746d	764c	773c										
INITLOAD(sysglobals)	***	308d	384d	509c												
KEYS(keys)		69d														
LIBRARIAN		72d	80c	94c	104c	118c	129c									
M8KSYS(ci)		2736d	2770c	2771c	2772c	2798c	2800c	2807c	2809c	2818c	2821c					
MINIT(asmr)		306d	341c	383c	731d	738c	740c									
MIU		1328d	1329d													
PRINTER(prtdvr)		453d	463c	496c	493c	495c	497c									
SRMMAM(srmmamodule)		40d	47d	28c												
SYSDEVS(sysdevs)		41d	179d	196c	203c											
UCSD_AM(ucsd_am)		241d	373d	377d	408d	433d	434c	485d	486c	529c	536c	552c	563c	575c	637c	
		41d	152c	176c	186c	214c										
buffer_busy																
IOLIB(general_4)		1247d	1658d	1662c	1663c											
buffer_cleanup																
GLE_HPGL(gle_hppl_out)		7057d	7127c	7348c	7374c	7411c	7458c	7477c								
buffer_data																
IOLIB(general_4)		1233d	1285d	1290c	1319c	1384c	1410c	1538c	1561c	1623c						
buffer_fudge																
GLE_HPGL(gle_hppl_out)		7010d	7061c													
GLE_HPGL(gle_hppl_in)	***	18010d														
buffer_mode																
GLE_GEN(gle_gen)	***	2222c														
GLE_HPGL(gle_hppl_out)	***	7546c														
GLE_RGL(gle_ras_out)	***	8421c														
GLE_TYPES(gle_types)	***	1030d														
buffer_reset																
IOLIB(general_4)		1230d	1296d	1320c	1384c											
buffer_space																
IOLIB(general_4)		1231d	1314d	1321c	1396c	1493c	1541c	1588c	1644c							
buffered_transfer																
AMIGO(amigodvr)		681c	686c	687c	722c	752c										
DISCHPIB(bkgnd)	***	45d														
buffer_size																
CRT(crt)		231d	243c	249c	256c											
buffsize																
INIT(misc)		484d	491c	503c	505c	516d	521c	525c	526c	535d	588c	590c	596c	599c	603c	
		610c	628c	637c	638c	640c	683c									
UCSD_AM(ucsd_am)		41d	176c	187c	190c	227c	257c									
buflength																
ASCII(asciimodule)		10d	21c	22c	265c											
bufparams																
ASCII(asciimodule)		18d	49c	85c	266c	295c	305c									
bufptr																
CS80(cs80dvr)		1095d	1116c	1215d	1250c	1268c	1269c	1278c	1283c	1295c	1296c	1307c	1308c	1318c		
ETU		423d	427c	428c	429c	431c	446c	449c	454c	458c	461c	462c	466c	474c	475c	
F9885(f9885dvr)		476c	478c	482c	606d	609c	613c	614c	615c	618c	619c	620c				
LIBRARIAN		82d	178c	213c	281c	282c										
LIBRARY(module)		2736d	2772c	2774c	2778c	2781c	2793c	2794c	2795c	2813c	2814c	2816c	2817c			
MINIT(xminit)		398d	403c	444c	447c											
		1386d	1481c	1544c	1630c	1724c	1759c	1800c								
bufrec																
ETU		422d	429c	430c	431c	436c	440c	443c	446c	448c	449c	452c	454c	476c	477c	
		478c	479c													
bufsize																
ASCII(asciimodule)		7d	14d	182c	190c	195c	202c	219c	222c	226c	239c	245c	248c			
BUBBLES(bubble)		30d	54d	74c	78c	81c	85c	96c								
EDRIVER(edriver)		39d	77d	115c	196c	199c	204c	225c	226c	267c	280c	288c	291c	309d	322c	
EPROMS(eproms)		9d	13d	51c	54c	61c	67c	69c	72c							
ETU		860d	938c	939c	940c	1043c	1050c	1051c								
INIT(initunits)		2140d	2149d	2153d	2157d	2229d										
INIT(misc)		542d	546c	547c	548c	555d	560c	561c	563c	564c	567c					
INITLOAD(bootdammodule)		545d	748d	757c	764c	773c										
INITLOAD(sysglobals)	***	63d														
M8KSYS(ci)		307d	335c	348c	353c	354c	359c	365c	367c	371c	376c	383c				
SRMMAM(srmmamodule)		42d	180d	196c	203c											
SYSDEVS(sysdevs)		373d	377c	408d	433d	434c	485d	486c								
buftype																
ETU		41d	42d													
buildchar																
KEYS(keys)		53d	503c	549c	553c											
buildcount																
KEYS(keys)		54d	503c	549c	550c											
burn																
ETU		43d	583c	596c												
burnit																
ETU		564d	620c	1082c	1098c	1104c	1110c									
burst_tfr																
KERNEL(iodeclarations)	***	562d														
bus_address																
CTABLE		1212d	1465c	1468c												
PRINTER(prtdvr)		69d	115c	221c	275c											
busaddr																
MINIT(asmr)		1331d	1335d	1342d	1346d											
busy																
A804XDVR(a804xdvr)		54d	135c													
BAT(bat)		42d	56c													
CRT(crt)	***	232d														
EDRIVER(edriver)		59d	218c	233c	247c											
UNITIO(uio)		118d	121c	122c												
button																
OGL_HPGLI(ogl_hppli)		18037d	18081c	18096c	18099c											
OGL_KNOB(ogl_knob)		18037d	18111c	18128c	18131c											
OGL_VARS(ogl_vars)	***	1141d														
GLE_HPGLI(ogle_hppl_in)	***	18227d														
LIB(ogl_lib)		20077d	20952d	20995c												
bx_bufptr																
AMIGO(amigodvr)		625c	768c	843c	846c	951c										
DISCHPIB(bkgnd)	***	49d														
bx_length																
AMIGO(amigodvr)		626c	748c													

[illegible]

[illegible]

[illegible]

char_width	***	\$T25c													
GLE_MARK(gle_smark)		4037c	4046c	4062c	4065c										
GLE_TEXT(gle_text)		1121d													
GLE_TYPES(gle_types)	***														
character															
CRT(crt)		192c	615c												
LOLIE(general_2)		840d	842c												
PRINTIE(prtdvr)		145d	232c												
SYSDEVS(sysdevs)	***	48d													
charde															
CRT(crt)		99d	145d												
GCR(crtb)	***	78d													
SYSDEVS(sysdevs)	***	94d													
chart															
AMIGC(amigodvr)	***	794d													
CONVERT(convert_txt)		45d	67c												
CRT(crt)	***	406d													
CS80(cs80dvr)	***	1215d													
DC_DRV(intdc)	***	498c													
DISCHP18(bkgnd)	***	49d													
DISCHP18(dischpb)		198d	198d	202d	263c	287d	357d	428c	440d						
F98S(f98sdvr)		82d	110d												
GCR(crtb)	***	276d													
GLE_KNOB(gle_knob_in)		18045d	18110d												
INIT(isr)		39d	46d	61d	79d										
INITLOAD(bootdammodule)		781d	786c	787c	788c										
INITLOAD(mini)	***	392d													
INITLOAD(sysglobals)		32d	205d												
KEYS(keys)	***	77d													
M68KSYS(ci)	***	311d													
MINIT(xminit)	***	1520c													
PRINTER(prtdvr)		68d	80d												
SRMAM(srmamodule)		53d	100d												
SRA(LRV)	***	2104c													
UCSD_AM(ucsd_am)	***	44d													
UNITIO(uio)		28d	30d	53d	74d										
charvar															
DGL_OUT(dgl_con'g_out)		11051d	11069c												
check															
CTABLE(ctr)		374d	657c	670c	671c	672c	679c	680c	681c	690c	691c	693c	702c	703c	704c
		706c	713c	714c	715c	724c	725c	726c	730c	732c	741c	742c	743c	744c	749c
		751c	764c	765c	766c	767c	783c	784c	785c	786c	808c	815c			
		43d	573c												
ETU		132d	161c	168c											
MOUSE(mouse)															
check2															
LIFDAM(lifmodule)		739d	753c	760c	780c										
checkchr															
MORESYS(mfs)		59d	126c	153c	185c	217c	257c	295c	328c	367c					
checkdsj															
AMIGC(amigodvr)		548d	778c	784d											
checkioresult															
MIUT		379d	494c	496c	498c	534c									
check_protectcode_swt_array															
SRMDAM(srmdammodule)		171d	490c	1061c											
checkstatus															
HINIT(hminit)		314d	372c	445c											
checkasoc															
LIBRARIAN		3308d	3480c	3501c											
checkcard															
ETU		637d	642c	648c	701c	781c									
checkctrl															
M68KSYS(ci)		315d	349c	368c											
checkentry															
LIFDAM(lifmodule)		805d	877c												
<hr/>															
checkeprom															
ETU		653d	729c	875c	1176c										
checkftitle															
LIFDAM(lifmodule)		220d	670c	969c	1066c	1087c	1164c								
checkkref															
LIBRARIAN		2279d	2641c												
checkloop															
HPHIL(hphil)		204d	236c	296c											
checkloopop															
HPHIL(hphil)	***	296c													
SYSDEV(Sysdevs)	***	306d													
checkname															
UCSD_DAM(ucsdmodule)		170d	172c	178c	210c	285c	303c	430c	446c						
checkrev															
INIT(ldr)		2438c	2456c												
INITLOAD(loader)		1085d	1578d	1623c											
SEGENTER(asm)	***	196c													
checkscode															
ETU		646d	659c	759c											
checkspace															
ETU		487d	494c	513c	1001c										
LIFDAM(lifmodule)		736d	810c	856c	893c										
UCSD_DAM(ucsdmodule)		190d	228c	233c											
checktextbuf															
LIBRARIAN		2241d	2662c	2669c	2676c	2682c	2689c	2710c							
checkvoiid															
LIFDAM(lifmodule)		307d	393c	501c	563c	592c	670c	968c	1066c	1085c					
chs															
SRMDAM(srmdammodule)		911d	942c												
c1															
ETU	***	25d													
LIBRARIAN	***	27d													
M68KSYS	***	1211d													
M68KSYS(ci)	***	30d													
ci_clr															
CS80(:s80)		390d	394c												
ci_cmd															
M68KSYS(c1)		63d	1063c	1094c	1100c	1101c	1105c	1107c	1109c	1110c	1143c	1151c	1152c	1154c	
ci_idie															
M68KSYS(c1)		62d	140c	142c	1098c	1173c									
c1_swit:h															
M68KSYS	***	1214c													
TAIL	***	38c													
cic															
CS80(:s80)		387d	393c	394c	395c										
cindex															
MOUSE(mouse)		65d	83c	95c	96c	103c	104c								
cinfo															
INIT(misc)	***	197J													
LIFDAM(lifmodule)		675c	619c												
SRMDAM(srmdammodule)		415c	417c	613c	616c	618c	620c	621c	625c	626c					
UCSD_DAM(ucsdmodule)		526c	558c												
cinfo0															
CRT(crt)		175d	692c												
cinfo1															
CRT(crt)	***	175d													
cinfo2															
CRT(crt)	***	175d													
cinfo3															
CRT(crt)	***	175d													
ciseoln															
M68KSYS(cil)		409d	413c	482c	489c	502c	512c	523c							

II-24


```

ck_display_init
  DGL_INQ(dgl_inq) 6043c 6069c
  DGL_POLY(dgl_poly) 20112c 20132c 20148c 20174c 20933c
  GEN(dgl_gen) 3040d 3081d
  LIB(dgl_lib) 20253c 20382c 20415c 20437c 20453c 20475c 20501c 20545c 20883c 20699c 20739c 20761c 20779c 20797c
  20824c 20859c 21030c

ck_for_graphics_board
  DGL_C_OUT(dgl_confg_out) 11192d 11270c

ck_locator_init
  GEN(dgl_gen) 3042d 3089d
  LIB(dgl_lib) 20271c 20860c 20894c 20969c 21073c

ck_system_init
  DGL_INQ(dgl_inq) 6042c 6068c 6169c
  DGL_POLY(dgl_poly) 20111c 20131c 20147c 20173c 20932c
  GEN(dgl_gen) 3038d 3073d
  LIB(dgl_lib) 20252c 20270c 20291c 20326c 20353c 20381c 20389c 20402c 20414c 20438c 20452c 20474c 20500c 20544c
  20682c 20698c 20738c 20760c 20778c 20796c 20823c 20858c 20893c 20968c 21029c 21072c 21095c 21204c
  21256c 21305c

ckind
  INIT(misc) *** 182d
  LIFDAM(lifmodule) *** 610c
  SRMDAM(srmamodule) *** 604c
  UCSD_DAM(ucsdmodule) *** 545c

cl
  MINIT(cs80ir) 710d 718c 719c 720c 721c 722c

clastdate
  INIT(misc) *** 190d
  LIFDAM(lifmodule) 573c 617c
  SRMDAM(srmamodule) 413c 436c 609c
  UCSD_DAM(ucsdmodule) 524c 554c

clasttime
  INIT(misc) *** 191d
  LIFDAM(lifmodule) 573c 617c
  SRMDAM(srmamodule) 414c 436c 609c
  UCSD_DAM(ucsdmodule) 525c 555c

cleandir
  LIFDAM(lifmodule) 359d 395c 972c 983c 1091c 1099c

cleanup
  LIFDAM(lifmodule) 212d 1158c 1171c 1174c
  TAPEBKUP 1229d 1274c
  UCSD_DAM(ucsdmodule) 137d 214c 296c 305c

clear
  CRT(crt) 315d 471c 472c 473c
  CTABLE 1241d 1253c 1255c 1257c 1259c 1261c
  GCRT(crtb) 224d 342c 343c 344c
  GLE_GEN(gle_gen) *** 2096c
  GLE_HPGL(gle_hppl_out) *** 7529c
  GLE_RGL(gle_ras_out) 8398c 8480c
  GLE_TYPES(gle_types) *** 1032d
  IOLTB(hpib_2) 1780d 1838d
  LIBRARIAN 3289d 3295c 3335c 3338c 3341c 3342c 3345c 3520c 3528c 3574c 3582c 3587c 3591c 3593c
  3599c 3602c 3608c 3610c

clear_and_escape
  F9885(f9885dvr) 94d 132c 198c 202c 209c 225c 245c

clear_display
  LIB(dgl_lib) 20055d 20539d

clear_hpib
  IOLTB(hpib_0) 1711d 1733d
  IOLTB(hpib_2) 1820c 1821c 1829c 1874c 1876c

clear_logs
  MINIT(cs80ir) 563d 701d 724c
  MINIT(qminit) 1173c

clear_logs_option_byte
  MINIT(qminit) 897d 1066c 1077c 1173c

clear_serial
  IOLTB(serial_0) 2217d 2294d

clear_unit
  AMIGO(amigodvr) 490d 582c
  CS80(cs80dvr) 999d 1230c 1366c
  F9885(f9885dvr) 85d 269c
  INITLOAD(mini) 395d 497c
  PRINTER(prtdvr) 108d 298c

clearanychar
  KEYS(keys) 349d 368c 516c 540c 545c 553c

clearchars
  GCRT(crtb) 226d 231c 233c 234c 235c

cleariohook
  CTABLE *** 1436c
  HPHIL(hphil) 369c 370c
  INIT *** 2544c
  INITLOAD(sysglobals) *** 273d
  KERNEL(general_0) *** 1244c
  M68KSYS(c1) *** 295c

clearline
  CRT(crt) *** 130d
  GCRT(crtb) *** 63d
  M68KSYS(c1) 68d 560c 563c
  SYSDEVS(sysdevs) *** 83d

clearlogs
  MINIT(cs80ir) 714d 720c

clearscr
  CRT(crt) *** 473c
  ETU *** 1279c
  GCRT(crtb) *** 344c
  INIT(misc) *** 167d
  M68KSYS(c1) *** 558c
  TAPEBKUP 1293c 1432c 1466c 1513c 1531c

clearscreen
  CRT(crt) *** 129d
  GCRT(crtb) *** 62d
  M68KSYS(c1) 67d 183c 558c 878c 1102c 1197c
  SYSDEVS(sysdevs) *** 82d

clearunit
  AMIGO(amigodvr) *** 575c
  BUBBLES(bubble) *** 107c
  CRT(crt) *** 416c
  CS80(cs80dvr) *** 1364c
  CTABLE(ctr) *** 842c
  EPROMS(eproms) 47c 78c
  F9885(f9885dvr) *** 268c
  GCRT(crtb) *** 286c
  INITLOAD(mini) *** 496c
  INITLOAD(sysglobals) *** 68d
  KEYS(keys) *** 84c
  M68KSYS(c1) 587c 963c
  MINIT(bminit) *** 79c
  MIUI *** 497c
  PRINTER(prtdvr) *** 297c
  SRMDAM(srmamodule) 188c 206c
  UNITIO(uio) *** 97c

clip_limits
  GLE_GEN(gle_gen) *** 2120c
  GLE_HPGL(gle_hppl_out) *** 7532c
  GLE_RGL(gle_ras_out) *** 8400c
  GLE_TYPES(gle_types) *** 1033c

clip_limits_xmax
  GLE_SCLIP(gle_sclip) *** 6018c
  GLE_TYPES(gle_types) *** 1131d

```



```

clip_limits_xmin
GLE_SCLIP(gle_sclip)      *** 6017c
GLE_TYPES(gle_types)      *** 1130d
clip_limits_ymax
GLE_SCLIP(gle_sclip)      *** 6020c
GLE_TYPES(gle_types)      *** 1133d
clip_limits_ymin
GLE_SCLIP(gle_sclip)      *** 6019c
GLE_TYPES(gle_types)      *** 1132d
clipping_support
DGL_HPGL(dgl_hppl)        *** 17264c
DGL_INQ(dgl_inq)          *** 6308c
DGL_RAS(dgl_raster)       *** 17595c
DGL_VARS(dgl_var)         *** 1107d
cliclear
CRT(crt)                   *** 522c
GCRT(crtb)                 *** 395c
KEYS(keys)                 *** 464c
SYSDEVS(sysdevs)           51d 542c 548c
clidisplay
CRT(crt)                   *** 526c
GCRT(crtb)                 *** 397c
KEYS(keys)                 *** 470c
PRINTER(prtdvr)           *** 188c
SYSLEVS(sysdevs)           51d
clicput
CRT(crt)                   *** 509c
GCRT(crtb)                 *** 381c
SYSDEVS(sysdevs)           51d 533c 552c 556c 560c 564c 573c
clishiftl
CRT(crt)                   *** 511c
GCRT(crtb)                 *** 383c
SYSDEVS(sysdevs)           51d 528c
clishiftr
CRT(crt)                   *** 517c
GCRT(crtb)                 *** 389c
SYSLEVS(sysdevs)           51d 572c
clock
CLOCK                      *** 182d
CLOCK(clock)               *** 32d
clockdata
CLOCK(clock)               *** 163d
SYSDEVS(sysdevs)           200d 205d 453d
clockdate
M68KSYS(c1)                 178d 255c 260c 267c 285c
clockcatetime
M68KSYS(c1)                 179d 283c 287c
clockfunc
CLOCK(clock)               *** 183d
SYSLEVS(sysdevs)           188d 205d 453d
clockinit
CLOCK                      *** 30d
clocktohook
A804XDVR(a804xdv)          *** 393c
CLOCK(clock)               53c 93c 103c 117c
SYSDEVS(sysdevs)           209d 460c 610c
clocktype
SYSDEVS(sysdevs)           206d 209d
clockop
A804XDVR(a804xdv)          *** 239d
SYSDEVS(sysdevs)           199d 206d 455d
clockops
A804XDVR(a804xdv)          239d 393c

clockreqhook
CLOCK(clock)               *** 177c
SYSDEVS(sysdevs)           208d 464c 467c 471c 475c 609c
clockreqtype
SYSDEVS(sysdevs)           205d 208d
clocktime
M68KSYS(c1)                 177d 256c 271c 280c 286c
closeall
ETU                          414d 1131c 1138c 1266c
closed_share_code
SRM_DRV(srm)               *** 164d
closeair
ETU                          333d 415c
closedirectory
ETU                          *** 341c
INITLDAD(sysglobls)         *** 136d
LIFDAM(lifmodule)           *** 1154c
SRMDAM(srmdammodule)        *** 1650c
UCSD_DAM(ucsdmodule)        *** 646c
closeown
M68KSYS(c1)                 328d 342c 358c 377c
closef
LIFDAM(lifmodule)           990d 1130c
closefile
ETU                          *** 405c
INIT(misc)                  *** 434c
INITLDAD(bootdammodule)     *** 721c
INITLDAD(loader)            *** 1391c
INITLDAD(sysglobls)         *** 135d
LIFDAM(lifmodule)           *** 1130c
SRMDAM(srmdammodule)        *** 1658c
UCSD_DAM(ucsdmodule)        *** 641c
closefiles
INIT(ldr)                   *** 2490c
INITLDAD(loader)            1090d 1385d 1561c 1647c
LIBRARIAN                    3007c 3687c 3703c
SEGMENTER(asm)              *** 214c
closeon
LIBRARIAN                    2957d 3006c 3031c 3079c 3105c 3684c 3687c 3702c
closeinfile
ETU                          397d 415c
closeit
UCSD_DAM(ucsdmodule)        404d 641c
closeout
LIBRARIAN                    3103d 3637c
closepack
SRMDAM(srmdammodule)        340c 927c
SRM_DRV(srm)                1033d 1681d
closest_error
DGL_RAS(dgl_raster)         17084d 17107c 17158c 17160c
closest_index
DGL_RAS(dgl_raster)         17086d 17106c 17161c 17164c
closetype
INIT(fs)                     744d 767d
clr_inhibit
DGL_HPGL(dgl_hppl)          17251d 17280c
DGL_RAS(dgl_raster)         17555d 17609c
clr_with_in
GLE_RGL(gle_ras_out)        8081d 8228c
clsize
INIT(misc)                  *** 185s
LIFDAM(lifmodule)           *** 570c
SRMDAM(srmdammodule)        *** 407c
UCSD_DAM(ucsdmodule)        *** 519c

```


cmap_address			8067d	8343c														
GLE_RGL(gle_ras_out)																		
cmap_def			8015d	8021d														
GLE_RGL(gle_ras_out)																		
cmd																		
R804XDVR(a804xdvr)			40d	42d	138d	139c	146d	147c	169d	171c	181d	197c	201c	206c	239d	241c		
BAT(bat)			67d	76c														
CLKK(clock)			163d	165c														
CS80(cs80)			355d	401d	408d	411c	418d	424d	428c	556d	576d	588c						
M68KSYS		***	28d															
SYSDSVS(sysdevs)			40d	185d	205d	208d	371d	380d	405d	428d	430c	437d	438c	445d	453d	455d		
cmd1																		
CS80(cs80)		***	240d															
cmd100																		
CS80(cs80)		***	265d															
cmd101																		
CS80(cs80)		***	265d															
cmd102																		
CS80(cs80)		***	265d															
cmd103																		
CS80(cs80)		***	265d															
cmd104																		
CS80(cs80)		***	266d															
cmd105																		
CS80(cs80)		***	266d															
cmd106																		
CS80(cs80)		***	266d															
cmd107																		
CS80(cs80)		***	266d															
cmd108																		
CS80(cs80)		***	267d															
cmd109																		
CS80(cs80)		***	267d															
cmd111																		
CS80(cs80)		***	242d															
cmd110																		
CS80(cs80)		***	267d															
cmd111																		
CS80(cs80)		***	267d															
cmd112																		
CS80(cs80)		***	268d															
cmd113																		
CS80(cs80)		***	268d															
cmd114																		
CS80(cs80)		***	268d															
cmd115																		
CS80(cs80)		***	268d															
cmd116																		
CS80(cs80)		***	269d															
cmd117																		
CS80(cs80)		***	269d															
cmd118																		
CS80(cs80)		***	269d															
cmd119																		
CS80(cs80)		***	269d															
cmd12																		
CS80(cs80)		***	243d															
cmd120																		
CS80(cs80)		***	270d															
cmd121																		
CS80(cs80)		***	270d															
cmd122																		
CS80(cs80)		***	270d															
cmd123																		
CS80(cs80)		***	270d															
cmd124																		
CS80(cs80)		***	271d															
cmd125																		
CS80(cs80)		***	271d															
cmd126																		
CS80(cs80)		***	271d															
cmd127																		
CS80(cs80)		***	271d															
cmd128																		
CS80(cs80)		***	272d															
cmd199																		
CS80(cs80)		***	244d															
cmd20																		
CS80(cs80)		***	245d															
cmd21																		
CS80(cs80)		***	245d															
cmd22																		
CS80(cs80)		***	245d															
cmd23																		
CS80(cs80)		***	245d															
cmd25																		
CS80(cs80)		***	246d															
cmd26																		
CS80(cs80)		***	246d															
cmd27																		
CS80(cs80)		***	246d															
cmd28																		
CS80(cs80)		***	247d															
cmd29																		
CS80(cs80)		***	247d															
cmd3																		
CS80(cs80)		***	240d															
cmd30																		
CS80(cs80)		***	247d															
cmd31																		
CS80(cs80)		***	247d															
cmd5																		
CS80(cs80)		***	241d															
cmd54																		
CS80(cs80)		***	253d															
cmd63																		
CS80(cs80)		***	255d															
cmd7																		
CS80(cs80)		***	241d															
cmd75																		
CS80(cs80)		***	258d															
cmd76																		
CS80(cs80)		***	259d															
cmd77																		
CS80(cs80)		***	259d															
cmd78																		
CS80(cs80)		***	259d															
cmd79																		
CS80(cs80)		***	259d															
cmd80																		
CS80(cs80)		***	260d															
cmd81																		
CS80(cs80)		***	260d															
cmd82																		
CS80(cs80)		***	260d															
cmd83																		
CS80(cs80)		***	260d															

II-27

cmd84	CS80(cs80)	***	261d																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																
-------	------------	-----	------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

cmdset_unit_0					
CS80(cs80)	248d	393c	427c	439c	579c
MINIT(cs80ir)	585c	638c			
TAPEBKUP(cs80tbr)	50c	65c			
cmdset_unit_1					
CS80(cs80)	***	248d			
cmdset_unit_10					
CS80(cs80)	***	250d			
cmdset_unit_11					
CS80(cs80)	***	250d			
cmdset_unit_12					
CS80(cs80)	***	251d			
cmdset_unit_13					
CS80(cs80)	***	251d			
cmdset_unit_14					
CS80(cs80)	***	251d			
cmdset_unit_15					
CS80(cs80)	***	251d	511c		
TAPEBKUP(cs80tbr)	***	234c			
cmdset_unit_2					
CS80(cs80)	***	248d			
cmdset_unit_3					
CS80(cs80)	***	248d			
cmdset_unit_4					
CS80(cs80)	***	249d			
cmdset_unit_5					
CS80(cs80)	***	249d			
cmdset_unit_6					
CS80(cs80)	***	249d			
cmdset_unit_7					
CS80(cs80)	***	249d			
cmdset_unit_8					
CS80(cs80)	***	250d			
cmdset_unit_9					
CS80(cs80)	***	250d			
cmdset_vol_0					
CS80(cs80)	***	256d	451c	580c	
MINIT(cs80ir)	***	586c			
TAPEBKUP(cs80tbr)	***	51c	66c		
cmdset_vol_1					
CS80(cs80)	***	256d			
cmdset_vol_2					
CS80(cs80)	***	256d			
cmdset_vol_3					
CS80(cs80)	***	256d			
cmdset_vol_4					
CS80(cs80)	***	257d			
cmdset_vol_5					
CS80(cs80)	***	257d			
cmdset_vol_6					
CS80(cs80)	***	257d			
cmdset_vol_7					
CS80(cs80)	***	257d			
cmdspare_block					
CS80(cs80)	***	241d			
MINIT(cs80ir)	***	864c			
cmdunload					
CS80(cs80)	***	258d			
TAPEBKUP(cs80tbr)	***	172c			
cmdwrite_file_mark					
CS80(cs80)	***	258d			
TAPEBKUP(cs80tbr)	***	149c			
cmdmd40					
CLOCK(clock)	88d	97c			

cmdmd41															
CLOCK(clock)	129d	147c													
cmdmdb7															
CLOCK(clock)	87d	96c													
cmdmf2															
CLOCK(clock)	***	135s													
cmdmf3															
CLOCK(clock)	134d	155c													
cmdmf4															
CLOCK(clock)	133d	154c													
cmdmf5															
CLOCK(clock)	132d	153c													
cmdmf6															
CLOCK(clock)	131d	149c													
cmdmf7															
CLOCK(clock)	130d	148c													
cname															
ETU	***	376c													
INIT(misc)	***	180d													
LIFDAH(lifmodule)	***	511c	566c	609c											
M88KSYS(ci)	***	822c													
MTUI	***	518c	519c												
SRMDAM(srmdammodule)	***	395c	404c	432c	452c	602c	997c	1004c							
UCSD_DAM(ucsdmodule)	***	485c	487c	515c	544c										
cnormal															
INIT(fs)	***	744d													
cnt															
DC_DRV(extdc)	***	203d													
DGL_C_IN(dgl_conf_in)	***	12043d	12049c	12089d	12097c										
DGL_C_OUT(dgl_conf_out)	***	11158d	11279c	11406d	11419c										
DGL_HPGL(dgl_hppl)	***	17028d	17052c	17072d	17077c	17098c	17100c	17106c	17108c	17114c	17116c				
GLE_HPGL(gle_hppl_out)	***	7165d	7182c	7183c	7184c	7185c	7186c	7187c	7188c	7222d	7242c	7243c	7244c	7245c	7246c
GLE_HPGL(gle_hppl_in)	***	7247c	7248c												
GLE_HPGL(gle_hppl_in)	***	18096d	18108c	18109c	18110c	18111c	18112c	18113c	18114c	18134d	18148c	18149c	18150c	18151c	18152c
GLE_HPGL(gle_hppl_in)	***	18153c	18154c	18226d	18254c	18255c	18265c	18291d	18301c	18302c	18303c	18304c	18305c		
GLE_RGL(gle_ras_out)	***	10486d	10493c												
L187dgi_lib)	***	8377d													
L187dgi_lib)	***	21201d	21217c	21218c	21253d	21264c									
cntrl															
INIT(misc)	***	174d													
KEYS(keys)	***	527c													
cntrlpausekey															
KEYS(keys)	***	64d	521c												
code															
LIBRARIAN	***	298d	1198c	1210c	1216c										
SEGMENTER(asm)	***	96d	98c												
SEGMENTER(segmenter)	***	38d													
code_space															
SEGMENTER(asm)	***	292d	297c	304c	306c	311c									
codecount															
LIBRARIAN	***	310d	480c	485c	1297c										
coded_addr															
AMIGO(csamigo)	***	339d	347c	351c	352c	356c	357c	387c	400c						
codefid															
M88KSYS(ci)	***	47d	631c	642c	1025c										
codefile															
INIT(misc)	***	722c	723c												
INITLOAD	***	1708c													
INITLOAD(loader)	***	1141c													
INITLOAD(sysglobals)	***	48d													
LIBRARIAN	***	2893c	3082c	3148c											
M88KSYS(ci)	***	103c	665c	996c											
codeindex															
LIBRARIAN	***	311d	481c	484c	1297c										


```

counts
MOUSE(mouse)          *** 159c 160c
SYSEVS(sysdevs)      ***
cp
INITLOAD(loader)      *** 916d
LIBKARIAN             1397c 1400c
cpen
GLE_RGL(gle_ras_out)  *** 8053d
cpsize
INIT(misc)            *** 183s
LIFDAM(lifmodule)     *** 508c 509c 513c 521c 569c 570c 611c 613c
M68KSYS(c1)           *** 824c
MIU1                  *** 521c
SRMDAM(srmdammodule)  *** 408c 605c
UCSD_DAM(ucsdmodule)  *** 488c 490c 517c 547c 548c
cptr
EDRIVER(edriver)      91d 199c 215c 217c 220c 230c 232c 235c
EPROMS(eproms)        36d 61c 68c
cpvol
ETU                    62d 230c 234c 243c 247c 254c 361c 362c 363c 893c 896c 897c
cpk
DGL_VARS(dgl_vars)    *** 1251d
GLE_SMARK(gle_smrk)   *** 5095d 5121c 5136c 5158c 5172c
LIB(dgl_lib)           20220d 20228c 20240c 20570c 20576c 20600c 20606c 21176c 21426c
cpy
DGL_VARS(dgl_vars)    *** 1250d
GLE_SMARK(gle_smrk)   *** 5097d 5122c 5137c 5159c 5173c
LIB(dgl_lib)           20221d 20229c 20241c 20571c 20577c 20601c 20607c 21177c 21427c
cpymsg
INITLOAD              *** 1792c
INITLOAD(loader)      *** 1619c
cr
INIT(misc)            *** 168d
KEYS(keys)            *** 447c 524c
create_new
UCSD_DAM(ucsdmodule)  182d 635c
create_temp_unitable
CTABLE(ctr)           *** 1492c
CTABLE(ctr)           *** 302d 821d
createdefs
INITLOAD(loader)      1419d 1560c
createfile
INIT(misc)            *** 450c
INITLOAD(sysglobals) *** 135d
LIFDAM(lifmodule)     *** 1128c
SRMDAM(srmdammodule)  *** 1655c 1732c
UCSD_DAM(ucsdmodule)  *** 635c
createLinkpack
SRMDAM(srmdammodule)  1202c 1235c 1391c
SRM_DRV               *** 1780d
SRM_DRV(srm)          *** 1055d
createpack
SRMDAM(srmdammodule)  1007c 1086c 1123c 1129c 1134c 1146c 1420c
SRM_DRV(srm)          1041d 1728d
createsysname
INITLOAD              1734d 1785c
creation_date
SRMDAM(srmdammodule)  435c 608c 1465c
SRM_DRV(srm)          282d 738d
cross_unit
CS80(cs80)            149d 294d
TAPEBKUP(cs80tboir)   775c 867c

```

```

crt
CRT                  *** 731d
CRT(crt)             *** 32d
GLE_RGL(gle_ras_out) *** 8390c
GLE_TYPES(gle_types) *** 1272d
crt_config
C_H00K              39d 143c
DGL_TOOLS(dgl_tools) 20017d 20034c
crt_reg
C_H00K              53d 144c
DGL_TOOLS(dgl_tools) 20029d 20046c
crt_reg_def
C_H00K              43d 53d
DGL_TOOLS(dgl_tools) 20021d 20029d
crt_tm_name
CTABLE(ctr)         358d 645c
crt_wtb
KERNEL(general_0)   795d 1234c
crtb
GCRT                 *** 580d
GCRT(crtb)          *** 32d
crtcmdwrd
CRT(crt)             51d 270d 280d 664d
GCRT(crtb)           98d 197d
crtcolumn
MIU1                 224d 246c 248c 253c 255c 260c 261c 269c
TAPEBKUP             1111d 1133c 1135c 1140c 1142c 1147c 1148c 1156c
crtcommand
CRT(crt)             263d 275c 276c 287c 288c 681c 682c
crtcon
CRT(crt)             93d 139d
GCRT(crtb)           *** 72d
SYSEVS(sysdevs)      *** 92d
crtconfigreg
CRT(crt)             *** 687c
INIT_LOAD(sysglobals) *** 284d
crtconsttype
CRT(crt)             94d 139d
GCRT(crtb)           *** 72d
SYSEVS(sysdevs)      *** 66d 92d
crtcontroladdr
CRT(crt)             89d 135d 676c 677c
GCRT(crtb)           *** 68c
SYSEVS(sysdevs)      *** 89d
crtcre
CRT(crt)             *** 120d
GCRT(crtb)           *** 53d
SYSEVS(sysdevs)      *** 76d 107d
crtctrl
CRT(crt)             *** 120d
GCRT(crtb)           *** 53d
SYSEVS(sysdevs)      *** 107d
crtdeb
CRT(crt)             544d 704c
GCRT(crtb)           413d 514c
crtfre
CRT(crt)             *** 110d
GCRT(crtb)           *** 43d
SYSEVS(sysdevs)      *** 68d 105d
crtidreg
CRT(crt)             173d 688c 692c

```


crtinfo														
CRT(crt)	132d	185c	666c	720c										
GCRT(crtb)	65d	502c												
M68KSYS(ci)	462c	493c	505c	1089c										
MINIT(mminit)	***	158c												
SYSDEVS(sysdevs)	***	108d												
crtinithook														
CRT(crt)	***	705c												
GCRT(crtb)	***	517c												
SYSDEVS(sysdevs)		125d	627c											
crtio														
CRT(crt)	428c	442c	443c											
GCRT(crtb)	298c	312c	313c											
INIT(initunits)	2147d	2244c	2250c											
KEYS(keys)	***	94s												
SYSDEVS(sysdevs)		376d	484d											
crtiohook														
CRT(crt)	***	702c												
GCRT(crtb)	***	516c												
SYSDEVS(sysdevs)		119d	486c	628c										
crtirec														
CRT(crt)		86d	132d											
GCRT(crtb)	***	65d												
SYSDEVS(sysdevs)		87d	108d											
crtkind5														
SYSDEVS(sysdevs)		113d	130d											
crtline														
MIUI	224d	246c	248c	255c										
TAPEBKUP	1111d	1133c	1135c	1142c										
crtllhook														
CRT(crt)	***	703c												
DGL_C_OUT(dgl_confg_out)	***	11068c	11185c	11186c										
GCRT(crtb)		515c												
KEYS(keys)		464c	470c	471c										
PRINTER(prtdvr)	***	188c												
SYSDEVS(sysdevs)		126d	528c	533c	542c	548c	552c	556c	560c	564c	572c	573c	582c	629c
crtllops														
CRT(crt)	***	502d												
GCRT(crtb)	***	370d												
SYSDEVS(sysdevs)		51d	52d	378d	487d									
crtlltype														
DGL_C_OUT(dgl_confg_out)	***	11023d												
SYSDEVS(sysdevs)		52d	126d											
crtmemaddr														
CRT(crt)		88d	134d	688c										
GCRT(crtb)	***	67d												
MINIT(mminit)	***	159c												
SYSDEVS(sysdevs)	***	89d												
crtptr														
MINIT(mminit)		149d	159c	162c										
crtregtype														
CRT(crt)		50d	263d											
GCRT(crtb)	***	97d												
crtscreen														
CRT(crt)		57d	58d											
crttype														
CRT(crt)	***	119d												
GCRT(crtb)	***	52d												
SYSDEVS(sysdevs)	***	106d												
crtword														
CRT(crt)	***	57d												
GCRT(crtb)	***	93d												
SYSDEVS(sysdevs)	***	47d												

crunch									
INITLOAD(sysglobals)	***	136d							
LIFDAM(lifmodule)	***	1151c							
SRMDAM(srmdammodule)	***	1794c							
UCSD_DAM(ucsdmodule)	***	647c							
crunchit									
UCSD_DAM(ucsdmodule)		567d	647c						
crunchv									
LIFDAM(lifmodule)		384d	1151c						
cs80									
CS80(cs80)	***	63d							
CS80(cs80dsr)	***	598d							
CS80(cs80dvr)	***	972d							
CTABLE		1408c	1564c	1595c	1599c				
CTABLE(brstuff)		993d	1009c						
CTABLE(ctr)		255d	572c	768c	787c	796c			
CTABLE(scanstuff)	***	1137c							
MINIT(cs80ir)	***	516d							
MINIT(qminit)	***	878d							
TAPEBKUP(cs80tdvr)	***	305d							
TAPEBKUP(cs80tbr)	***	14d							
cs80_devid									
CS80(cs80dvr)		994d	1021c	1062c	1064c				
cs80_devtype									
CS80(cs80dvr)		993d	1020c	1059c					
cs80_hardvols									
CS80(cs80dvr)		995d	1022c	1067c	1069c				
cs80_mp									
CS80(cs80dvr)		996d	1023c	1071c					
cs80_tm_name									
CTABLE(ctr)		365d	768c	787c					
cs80dav									
CTABLE(scanstuff)		1041d	1133d	1137c					
cs80disc_pp									
CTABLE(ctr)	***	572c							
CTABLE(options)	***	186d							
cs80dsr									
CS80(cs80dsr)	***	595d							
CS80(cs80dvr)	***	972d							
MINIT(qminit)	***	878d							
TAPEBKUP(cs80tdvr)	***	305d							
cs80dt									
CTABLE		1214d	1372d	1380c	1381c	1568c			
CTABLE(scanstuff)		1042d	1054d	1134d	1138c	1141c			
cs80dvr									
CS80(cs80dvr)	***	969d							
cs80hardvols									
CTABLE		1216d	1374d	1380c	1383c	1568c	1579c	1762s	1763s
CTABLE(scanstuff)		1043d	1056d	1135d	1138c	1143c			
cs80id									
CTABLE		1215d	1373d	1380c	1568c	1576c	1746s	1756s	
CTABLE(scanstuff)		1042d	1055d	1134d	1138c	1142c			
cs80init									
CS80	***	30d							
cs80io									
CS80(cs80dvr)		987d	1351d						
cs80ir									
MINIT(cs80ir)	***	513d							
MINIT(qminit)	***	878d							
cs80mp									
CTABLE		1375d	1380c	1383c					
CTABLE(scanstuff*)		1043d	1057d	1135c	1138c	1144c			
cs80tape_msus									
CTABLE		1205d	1382c	1444c	1594c	1598c			

[illegible]

current_cursor_state			
GLE_HPGL(gle_hpgl_out)	***	7499c	7709c
GLE_TYPES(gle_types)	***	1135d	
LIB(dgl_lib)		20231c	20236c
current_cursor_x			
GLE_TYPES(gle_types)	***	1136d	
LIB(dgl_lib)		20232c	20237c
current_cursor_y			
GLE_TYPES(gle_types)	***	1137d	
LIB(dgl_lib)		20232c	20238c
current_drawing_mode			
GLE_TYPES(gle_types)	***	1148d	
LIB(dgl_lib)		20979c	
current_echo_type			
DGL_HPGL(dgl_hpgl)	***	18061c	
DGL_KNOB(dgl_knob)	***	18062c	
DGL_VARS(dgl_vars)	***	1325d	
GEN(dgl_gen)	***	3257c	
LIB(dgl_lib)		20150c	20178c
current_fill_index			
GLE_TYPES(gle_types)	***	1147d	
current_linestyle			
GLE_HPGL(gle_hpgl_out)	***	7438c	
GLE_SHARK(gle_shark)	***	5116c	
GLE_TYPES(gle_types)	***	1141d	
LIB(dgl_lib)		20976c	
current_linestyle_mode			
GLE_HPGL(gle_hpgl_out)	***	7440c	
GLE_SHARK(gle_shark)	***	5114c	
GLE_TYPES(gle_types)	***	1144d	
LIB(dgl_lib)		20978c	
current_linestyle_pattern			
GLE_HPGL(gle_hpgl_out)	***	7441c	
GLE_SHARK(gle_shark)	***	5117c	
GLE_TYPES(gle_types)	***	1142d	
LIB(dgl_lib)		20974c	
current_linewidth			
GLE_TYPES(gle_types)	***	1149d	
LIB(dgl_lib)		20975c	
current_patterr_length			
GLE_HPGL(gle_hpgl_out)	***	7439c	
GLE_SHARK(gle_shark)	***	5115c	
GLE_TYPES(gle_types)	***	1143d	
LIB(dgl_lib)		20977c	
current_polygon_blue			
GLE_RGL(gle_ras_out)	***	8191c	
GLE_TYPES(gle_types)	***	1154d	
current_polygon_color			
GLE_TYPES(gle_types)	***	1151d	
current_polygon_green			
GLE_RGL(gle_ras_out)	***	8190c	
GLE_TYPES(gle_types)	***	1153d	
current_polygon_red			
GLE_RGL(gle_ras_out)	***	8189c	
GLE_TYPES(gle_types)	***	1152d	
current_pos_x			
GLE_HPGL(gle_hpgl_out)		7346c	7372c 7707c
GLE_SHARK(gle_shark)	***	5121c	
GLE_TYPES(gle_types)	***	1110d	
LIB(dgl_lib)		20228c	

current_pos_y									
GLE_HPGL(gle_hpgl_out)		7347c	7373c	7708c					
GLE_SHARK(gle_shark)	***	5122c							
GLE_TYPES(gle_types)	***	1111d							
LIB(dgl_lib)		20229c							
current_record									
SRM_DRV(srm)	***	577d							
current_timeout									
PRINTER(prtdvr)		76d	219c	233c	241c	251c	282c		
current_vol									
CTABLE(ctr)		313d	314d	604d	609c	615d	617c		
current_vu									
CS80(cs80)	***	208d							
CS80(cs80dsr)	***	763c							
TAPEBKUP(cs80tdvdr)		653c	886c						
current_crt									
CRT(crt)	***	708c							
C_HODK	***	142c							
DGL_C_OUT(dgl_confg_out)	***	1183c	11286c	11288c	11296c	11302c	11332c	11335c	11342c 11346c
GCRT(crtb)	***	520c							
SYSDEVS(sysdevs)		130d	631c						
currenttif									
CS80(cs80)	***	116d							
MINIT(qminit)	***	914c							
curseadr									
CRT(crt)	***	270d	272c	273c	274c	275c	276c	664d	678c 679c 680c 681c 682c
GCRT(crtb)	***	197d							
cursor									
GLE_GEN(gle_gen)	***	2180c							
GLE_HPGL(gle_hpgl_out)	***	7544c							
GLE_RGL(gle_ras_out)	***	8417c							
GLE_TYPES(gle_types)	***	1034d							
LIB(dgl_lib)		20162d	20180c	20197c	20207c	21284c			
cursor_color									
DGL_VARS(dgl_vars)	***	1331d							
LIB(dgl_lib)		20990c	21448c						
cursor_size_x									
DGL_VARS(dgl_vars)	***	1328d							
cursor_size_y									
DGL_VARS(dgl_vars)	***	1329d							
cursor_x									
GLE_RGL(gle_ras_out)	***	8047d							
cursor_y									
GLE_RGL(gle_ras_out)	***	8048d							
cursoraddr									
GCRT(crtb)		107d	481c	482c					
cursormask									
CRT(crt)	***	107d	153d						
GCRT(crtb)	***	86d							
SYSDEVS(sysdevs)	***	100d							
curse									
CRT(crt)		566c	614c	619c	627c	634c	649c		
GCRT(crtb)	***	429c	451c	456c	466c	472c			
SYSDEVS(sysdevs)	***	60d							
cursey									
CRT(crt)		566c	614c	619c	628c	634c	649c		
GCRT(crtb)	***	429c	451c	456c	466c	472c			
SYSDEVS(sysdevs)	***	60d							
cvtl									
ETU		63d	241c	242c	247c	274c	276c	281c	363c 376c 897c 899c 933c
cvidatetime									
LIFDAH(lifmodule)		548d	573c	574c	617c				

data		81c																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													
------	--	-----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--


```

defblocks
  LIBRARIES
    2011d 2116c 2117c 2118c 2119c
defdelta
  INITLOAD(loader)
    1476d 1479c 1484c 1486c
defective_track
  AMIG(csamigodr)
    *** 814c
  AMIG(csamigo)
    *** 77d
deferraction
  INIT(misc)
    *** 237c
  INITLOAD(sysglobals)
    *** 277d
  SYSDEVS(sysdevs)
    500c 505c
define_color_map
  GLE_CEN(gle_gen)
    *** 2198c
  GLE_HPGL(gle_hppl_out)
    *** 7550c
  GLE_KGL(gle_ras_out)
    *** 8280c
  GLE_TYPES(gle_types)
    *** 1035d
define_drawing_mode
  GLE_CEN(gle_gen)
    *** 2204c
  GLE_HPGL(gle_hppl_out)
    *** 7549c
  GLE_KGL(gle_ras_out)
    *** 8418c
  GLE_TYPES(gle_types)
    *** 1036d
define_aseword
  LIBRARIES
    1149d 1221c 1230c
defneword
  LIBRARIES
    494d 1144c 1207c
deflimit
  LIBRARIES
    1165d 1169c 1170c 1187c
defmodptr
  INITLOAD(loader)
    1179d 1182c 1183c 1185c 1187c 1188c 1189c
defptr
  INITLOAD(loader)
    1248d 1262c 1264c 1265c 1269c 1276c
  LIBRARIES
    1484d 1572c 1574c 1577c 1578c 1579c
  MINIT(xminit)
    1411d 1447c 1565c 1644c 1645c 1651c 1747c 1748c 1754c 1781c 1785c
defsize
  CTABLE(ctr)
    *** 395c
  INITLOAD(loader)
    958d 1014d 1188c 1258c 1259c 1260c 1262c 1264c 1265c 1310c 1349c 1432c 1467c
  LIBRARIES
    375c 1169c 1327c 1560c 2019c 2043c 2114c 2115c 2116c
  SEGMENTER(asm)
    *** 123c
defsou
  LIBRARIES
    49d 2114c 2974c 3577c 3630c
deftable
  INITLOAD(loader)
    *** 1172d
  MINIT(xminit)
    1409d 1448c 1651c 1754c 1787c 1792c
deftablelength
  INITLOAD(loader)
    *** 1175d
des_to_rad
  DGL_PLY(dgl_poly)
    20042d 20899c 20993c
del
  INIT(misc)
    *** 173d
  KEYS(keys)
    478c 509c
  MINIT(cs801r)
    742d 750c 751c 752c 753c 754c
  MORESYS(mfs)
    53d 63c 124c 151c 183c 215c 255c 293c 326c 365c
del1
  SRMDAM(srmdammodule)
    217d 225c 665d 674c 684c 689c 691c 697c
del2
  SRMDAM(srmdammodule)
    218d 226c 666d 674c 689c
delay7t
  AB04XOVR(a804xdvr)
    272c 299c 317c
  SYSDEVS(sysdevs)
    *** 213d
delay_timer
  DISCHP1B(dischp1b)
    216d 281c 340c 344c
delay1
  AB04XOVR(a804xdvr)
    263c 291c
  SYSDEVS(sysdevs)
    *** 213d
delentry
  UCSD_DAM(ucsdmodule)
    379d 401c 423c 433c
delim
  SRMDAM(srmdammodule)
    212d 222c 224c 229c 251c 266c 277c 279c
delta2
  LIBRARIES
    2367d 2453c 2454c 2494c
delta_x_21
  DGL_PLY(dgl_poly)
    20275d 20287c 20297c 20298c 20307c 20308c
delta_x_31
  DGL_PLY(dgl_poly)
    20277d 20290c 20298c 20308c
delta_x_43
  DGL_PLY(dgl_poly)
    20279d 20293c 20297c 20307c
delta_y_21
  DGL_PLY(dgl_poly)
    20276d 20288c 20297c 20298c 20307c 20308c
delta_y_31
  DGL_PLY(dgl_poly)
    20278d 20291c 20298c 20308c
delta_y_43
  DGL_PLY(dgl_poly)
    20280d 20294c 20297c 20307c
dend
  LIFDAM(lifmodule)
    200d 284c 415c 869c
denom
  DGL_PLY(dgl_poly)
    20274d 20297c 20300c 20301c
density
  DGL_HPGL(dgl_hppl)
    17227d 17228d 17229d 17230d 17231d 17232d 17233d 17234d 17235d 17236d 17237d 17238d 17239d 17240d
  DGL_HPGL(dgl_hppl)
    17241d 17242d
  DGL_HPGL(dgl_hppl)
    *** 6078c
  DGL_PLY(dgl_poly)
    20156c 20157c 20184c
  DGL_RAS(dgl_raster)
    17562d 17563d 17564d 17565d 17566d 17567d 17568d 17569d 17570d 17571d 17572d 17573d 17574d 17575d
  DGL_VARS(dgl_vars)
    *** 1033d
dentry
  INITLOAD(bootdammodule)
    613d 665c 668c 671c 673c 674c 711c
  LIFDAM(lifmodule)
    201d 300c 418c 419c 421c 425c 428c 434c 437c 438c 453c 457c 465c 466c
    534c 535c 597c 598c 601c 602c 607c 639c 640c 646c 664c 807c 820c 855c
    860c 878c 915c 916c 935c 936c 937c 938c 940c 941c 942c 943c 944c
    945c 954c 956c 999c 1002c 1009c 1021c 1023c 1026c 1028c 1032c 1038c 1050c 1052c 1053c
    1054c 1076c 1077c 1078c 1093c 1094c 1095c 1165c
deovblk
  INITLOAD(loader)
    *** 833d
  LIBRARIES
    3108c 3159c
  MORESYS(csi)
    *** 776d
  UCSD_DAM(ucsdmodule)
    98c 233c 361c 488c 490c 517c 518c
describe
  CS80(cs80)
    344d 485d 496c
  CS80(cs80dsr)
    *** 909c
  CS80(cs80dvr)
    *** 1053c
  HPHIL(hphil)
    179d 244c
  MINIT(qminit)
    968c 1267c
  TAPEBUP(cs80tbdrv)
    *** 442c
describe_bytes
  CS80(cs80)
    344d 485d 494c
  CS80(cs80dsr)
    777d 909c 913c
  CS80(cs80dvr)
    1030d 1053c 1057c
  MINIT(qminit)
    891d 908c 921c 968c 973c 1002c 1016c 1060c 1162c 1192c 1267c
  TAPEBUP
    1248c 1299c 1316c 1440c 1470c
  TAPEBUP(cs80tbdrv)
    312d 442c 481c 492c
describe_type
  CS80(cs80)
    92d 344d 485d
  CS80(cs80dsr)
    *** 777d
  CS80(cs80dvr)
    *** 1030d
  MINIT(qminit)
    *** 891c
  TAPEBUP(cs80tbdrv)
    *** 312d
describe_data
  HPHIL(hphil)
    96c 189c

```


descrip		101c	103c	104c	107c	109c	110c	114c	115c	116c	118c	121c	196c	257c
HPHIL(hphil)		88c	93c	93c	159c	160c								
MOUSE(mouse)		331d												
SYSDEV5(sysdevs)	***													
descriprec														
SYSDEV5(sysdevs)		311d	331d											
dest														
LISRRARIAN		546d	914c											
destaddr														
SRM_DRV(srm)	***	300d												
destfid														
SRM_DRV(srm)		1038d	1705d	1718c										
destination_blk_addr														
TAPEBKUP(cs80tbr)		34d	210d	248c										
destination_block_address														
TAPEBKUP		1240c	1263c	1345c										
TAPEBKUP(cs80tbdrv)		334d	901c											
destination_block_size														
TAPEBKUP		1221d	1285c	1319c	1416c									
destination_file_id														
SRM_DRV(srm)		476d	1718c											
destination_is_a_7914														
TAPEBKUP		1220d	1237c	1259c	1315c	1374c								
destination_is_a_tape														
TAPEBKUP		1219d	1231c	1243c	1255c	1268c	1280c	1318c	1323c	1339c	1371c	1379c	1413c	
destination_lun														
TAPEBKUP		1217d	1307c	1308c	1310c	1314c								
destination_offset														
SRM_DRV(srm)		477d	1719c											
destination_size														
TAPEBKUP		1222d	1250c	1268c	1282c	1284c	1320c	1323c	1381c	1382c	1415c			
destination_uep														
TAPEBKUP		1232c	1247c	1248c	1256c	1269c	1286c	1308c	1309c	1312c	1315c	1372c	1387c	1417c
TAPEBKUP(cs80tbdrv)		332d	742c	901c										
TAPEBKUP(cs80tbr)		33d	210d	244c	245c									
destoff														
SRM_DRV(srm)		1039d	1706d	1713c										
dev														
AMIGO(amigodrv)		506d	511c	513c	520c	534c								
CTABLE(brstuff)		956d	1002c	1005c										
MIUI		319d	321c											
dev_addr														
L18(dgl_lib)		20025d	20038d	21245d	21264c	21278c	21295d	21317c	21328c					
dev_buf_ptr														
GLE_TYPES(gle_types)		1065d	1068d	1215d	1218d									
dev_dep_stuff														
DGL_C_IN(dgl_conf_in)		12034c	12054c											
DGL_C_OUT(dgl_conf_out)		11056c	11059c	11281c	11394c	11427c								
DGL_INQ(dgl_inq)		6198c	6214c											
DGL_RAS(dgl_raster)		17186c	17441c	17592c										
GLE_GEN(gle_gen)		2132c												
GLE_GEN(gle_gen)	***	3030c												
GLE_GEN(gle_gen)	***	7045c												
GLE_GEN(gle_gen)		7100c	7121c	7172c	7229c	7335c	7361c	7404c	7433c	7471c	7490c	7518c		
GLE_KNOB(gle_knob_in)		18083c	18119c	18186c	18197c	18216c								
GLE_RGL(gle_ras_out)		8181c	8231c	8296c	8326c	8381c								
GLE_TYPES(gle_types)		1072d	1222d											
dev_name														
MIUI		106d	193c											
dev_name_type														
MIUI		103d	107d											
dev_type														
CTABLE(brstuff)		948d	956d	987d										
devaddr														
HPHIL(hphil)		52d	343c	349c										


```

device.msus
CTABLE
device_name
LIB(dgl_lib)
device_table
RHIG0(amigodvr)
device_table_type
RHIG0(amigodvr)
device_work_area
DGL_C_OUT(dgl_confg_out)
deviceaddress
GLE_RGL(gle_ras_out)
devicename
MINI1(qminit)
MINI1
TAPEBKUP(cs80tbdvr)
devicerec
SYSDEV5(sysdevs)
devicetype
DGL_RAS(dgl_raster)
GLE_RGL(gle_ras_out)
devd
RHIG0(amigodvr)
RHIG0(csamigo)
CS80(cs80dsr)
CS80(cs80dvr)
CTABLE(ctr)
INIT1(initunits)
INITLOAD
INITLOAD(sysglobs)
MINI1(hminit)
PRINTER(prtdvr)
TAPEBKUP
devstate
HAPHIL(hphl)
MOUSE(mouse)
SYSDEV5(sysdevs)
devtype
CS80(cs80dvr)
dfirstblk
INITLOAD(loader)
LIBRARIAN
M68KSYS(cil)
UCSD_DAM(ucsdmodule)
dfkind
INITLOAD(loader)
LIBRARIAN
M68KSYS(cil)
UCSD_DAM(ucsdmodule)
dgl_lib
LIB(dgl_lib)
dgl_background_index
DGL_HPGL(dgl_hppl)
DGL_RAS(dgl_raster)
DGL_VARS(dgl_vars)
LIB(dgl_lib)
dgl_char_height
DGL_INQ(dgl_inq)
DGL_VARS(dgl_vars)
LIB(dgl_lib)
dgl_char_width
DGL_INQ(dgl_inq)
DGL_VARS(dgl_vars)
LIB(dgl_lib)
dgl_confg_in
DGL_C_IN(dgl_confg_in)
LIB(dgl_lib)
dgl_confg_out
DGL_C_OUT(dgl_confg_out)
DGL_KNOB(dgl_knob)
LIB(dgl_lib)
dgl_current_color
DGL_INQ(dgl_inq)
DGL_POLY(dgl_poly)
DGL_VARS(dgl_vars)
LIB(dgl_lib)
dgl_current_color_model
DGL_INQ(dgl_inq)
DGL_RAS(dgl_raster)
DGL_VARS(dgl_vars)
LIB(dgl_lib)
dgl_current_linestyle
DGL_INQ(dgl_inq)
DGL_POLY(dgl_poly)
DGL_VARS(dgl_vars)
LIB(dgl_lib)
dgl_current_linewidth
DGL_INQ(dgl_inq)
DGL_POLY(dgl_poly)
DGL_VARS(dgl_vars)
LIB(dgl_lib)
dgl_current_polygon_angle
DGL_POLY(dgl_poly)
DGL_VARS(dgl_vars)
LIB(dgl_lib)
dgl_current_polygon_color
DGL_INQ(dgl_inq)
DGL_POLY(dgl_poly)
DGL_VARS(dgl_vars)
LIB(dgl_lib)
dgl_current_polygon_crosshatch
DGL_POLY(dgl_poly)
DGL_VARS(dgl_vars)
LIB(dgl_lib)
dgl_current_polygon_density
DGL_POLY(dgl_poly)
DGL_VARS(dgl_vars)
LIB(dgl_lib)
dgl_current_polygon_edge
DGL_POLY(dgl_poly)
DGL_VARS(dgl_vars)
LIB(dgl_lib)
dgl_current_polygon_linestyle
DGL_INQ(dgl_inq)
DGL_POLY(dgl_poly)
DGL_VARS(dgl_vars)
LIB(dgl_lib)
dgl_current_polygon_spacing
DGL_POLY(dgl_poly)
dgl_current_polygon_style
DGL_INQ(dgl_inq)
DGL_POLY(dgl_poly)
DGL_VARS(dgl_vars)
LIB(dgl_lib)

```



```

dgl_current_timing_mode
DGL_INQ(dgl_inq)      *** 6369c
DGL_VARS(dgl_vars)    *** 1092d
LIB(dgl_lib)          20391c 21169c 21447c
dgl_cursor
LIB(dgl_lib)          20216d 21284c
dgl_gen
DGL_HPGLI(dgl_hpqli)  *** 18015d
DGL_INQ(dgl_inq)      *** 6031d
DGL_KNOB(dgl_knob)    *** 18015d
DGL_POLY(dgl_poly)    *** 20035d
DGL_RAS(dgl_raster)   *** 17019d
GEN(dgl_gen)          *** 3006d
LIB(dgl_lib)          *** 20103d
dgl_hpqli
DGL_C_OUT(dgl_cfg_out) *** 11020d
DGL_HPGLI(dgl_hpqli) *** 17002d
dgl_hpqli_init
DGL_C_OUT(dgl_cfg_out) *** 11447c 11487c
DGL_HPGLI(dgl_hpqli) *** 17006d 17220d
dgl_hpqli
DGL_C_IN(dgl_cfg_in)  *** 12019d
DGL_HPGLI(dgl_hpqli) *** 18002d
dgl_hpqli_init
DGL_C_IN(dgl_cfg_in)  *** 12117c
DGL_HPGLI(dgl_hpqli) *** 18006d 18123d
dgl_libody
LIB(dgl_lib)          *** 20110d
dgl_inq
DGL_INQ(dgl_inq)      *** 6002d
dgl_knob
DGL_C_IN(dgl_cfg_in)  *** 12018d
DGL_KNOB(dgl_knob)    *** 18002d
dgl_knob_init
DGL_C_IN(dgl_cfg_in)  *** 12058c
DGL_KNOB(dgl_knob)    *** 18006d 18169d
dgl_lib
DGL_POLY(dgl_poly)    *** 20038d
LIB(dgl_lib)          *** 20012d
dgl_poly
DGL_POLY(dgl_poly)    *** 20008d
dgl_polygon_color_current
DGL_POLY(dgl_poly)    *** 20098c 20120c 20950c
DGL_VARS(dgl_vars)    *** 1146d
LIB(dgl_lib)          20420c 21166c
dgl_raster
DGL_C_OUT(dgl_cfg_out) *** 11019d
DGL_RAS(dgl_raster)   *** 17002d
dgl_raster_init
DGL_C_OUT(dgl_cfg_out) *** 11354c
DGL_RAS(dgl_raster)   *** 17006d 17544d
dgl_scaled_draw
LIB(dgl_lib)          *** 20865c
dgl_scaled_move
LIB(dgl_lib)          *** 20635c
dgl_tools
DGL_C_IN(dgl_cfg_in)  *** 12015d
DGL_C_OUT(dgl_cfg_out) *** 11016d
DGL_TOOLS(dgl_tools) *** 20003d
LIB(dgl_lib)          *** 20109d

```

```

dgl_types
DGL_HPGLI(dgl_hpqli) *** 17010d
DGL_HPGLI(dgl_hpqli) *** 18010d
DGL_INQ(dgl_inq)      *** 6004d
DGL_KNOB(dgl_knob)    *** 18010d
DGL_POLY(dgl_poly)    *** 20010d
DGL_RAS(dgl_raster)   *** 17010d
DGL_VARS(dgl_vars)    *** 52d
GEN(dgl_gen)          *** 3008d
LIB(dgl_lib)          *** 20014d
TYPES(dgl_types)     *** 1002d
dgl_vars
DGL_C_IN(dgl_cfg_in)  *** 12016d
DGL_C_OUT(dgl_cfg_out) *** 11021d
DGL_HPGLI(dgl_hpqli) *** 17011d
DGL_HPGLI(dgl_hpqli) *** 18011d
DGL_INQ(dgl_inq)      *** 6029d
DGL_KNOB(dgl_knob)    *** 18011d
DGL_POLY(dgl_poly)    *** 20034d
DGL_RAS(dgl_raster)   *** 17011d
DGL_VARS(dgl_vars)    *** 50d
GEN(dgl_gen)          *** 3054d
LIB(dgl_lib)          *** 20101d
di
LIFDAM(lifmodule)    582d 594c 624c 625c
diag_unit
MINIT(qminit)        1102d 1124c 1129c 1136c 1141c
diagnostic_release_required
CS80(cs80)           *** 159d
CS80(cs80dsr)         *** 710c
TAPEBKUP(cs80tbdrv)   *** 592c 824c
diagnostic_request
CS80(cs80)           *** 183d 312d 326d
CS80(cs80dsr)         *** 717c
TAPEBKUP(cs80tbdrv)   *** 599c 831c
diagnostic_result
CS80(cs80)           *** 156d 299d
CS80(cs80dsr)         *** 670c
TAPEBKUP(cs80tbdrv)   *** 550c 782c
diagnostic_to_be_run
MINIT(qminit)        894d 1063c 1071c 1072c 1122c
digit
GEN(dgl_gen)          3151d 3157c 3162c 3163c
GLE_UTILS(gle_utils) 21047d 21054c 21055c 21076d 21096c 21097c
digitize_enabled
GLE_KNOB(gle_knob_in) 18019d 18153c 18188c 18199c 18259c
index
LIFDAM(lifmodule)    200d 300c 414c 418c 419c 424c 432c 459c 460c 637c 642c 656c 657c 664c
723c 730c 731c 767c 768c 789c 790c 812c 825c 858c 865c 866c 878c 879c
890c 932c 945c 997c 999c 1008c 1020c 1050c 1053c 1070c 1095c
dir
LIFDAM(lifmodule)    79d 246c 251c 285c 300c 315c 317c 319c 352c 365c 367c 375c 407c 418c
457c 466c 528c 530c 535c 594c 597c 637c 639c 664c 695c 698c 708c 709c
713c 866c 876c 945c 999c 1009c 1032c 1050c 1053c 1076c 1078c 1095c 1122c 1144c
1182c
SRM_DRV(srm)          1298d 1309c 1325c 1346c
UCSD_DAM(ucsdmodule) 53d 60c 74c 87c 145c 149c 153c 155c 215c 221c 225c 226c 256c 257c
260c 272c 276c 277c 288c 293c 309c 317c 333c 339c 340c 360c 362c 373c
389c 390c 394c 411c 421c 451c 453c 468c 477c 497c 498c 500c 503c 504c
516c 538c 590c 613c 617c 621c 623c 630c 657c 675c 676c
dir_is_dvrttemp
SRM_DAM(srdammodule) 385d 426c
dir_len
MINIT(qminit)        400d 435d

```


II-44

[illegible]

doblankcheck															
ETU	1155d	1253c													
docat	580d	1152c													
LIFDAM(lifmodule)															
dochar	1373d	1400c	1404c												
docheckops	163d	177c													
CLOCK(clock)															
doconfigure	709d	872c	1173c	1236c	1251c										
ETU															
doctrl	402d	702c													
GCRT(crtb)	272d	516c													
doedit	3297d	3631c													
LIBRARIAN															
doinit	123d	227c													
NONUSKBD2															
doit	92d	96c	100c	102c	103c	105c	106c	108c							
A804XQVR(a804xdvr)	38d	403d													
SYSDEVS(sysdevs)															
dokbdio	72d	582c													
KEYS(keys)															
dokey	326d	330c	331c	334c	340c										
A804XQVR(a804xdvr)	355d	373d	422c												
KEYS(keys)															
domakedirectory	482d	1155c													
LIFDAM(lifmodule)															
dominant	8077d	8239c	8282c												
GLE_RGL(gle_ras_out)															
dominate_mode															
OGI_VARS(dgl_vars)	***	1201d													
doopen	71d	74d													
A804XDVR(a804xdvr)	414d	457c	468c	480c	500c	515c	523c								
DC_ORV(intdc)	20557d	20587c	20581c	20587c											
OGI_POLY(dgl_poly)	99d	118c	162c	169c	187c	188c	271c	273c	274c	287c	288c	289c			
EDRIVER(edriver)	41d	83c	92c	95c	100c	101c									
EPROM5(eproms)	656d	665c	684c	689c	692c	711d	848d	950c	956c	969c	1025c	1044c	1056c	1060c	
ETU	1063c	1113c	1122c	302d	316c										
HPHIL(hphil)	71d	96d	282d												
INIT(ldr)	2324d	2334c	2343c	2347c											
INITLOAD	1712d	1721c	1723c												
KEYS(keys)	375d	382c	389c	395c	399c	403c	415c	425c	459c	474c	480c	499c	515c	519c	
LIBRARIAN	525c	538c	550c	551c	561c										
	152d	184c	198c	201c	1483d	1555c	1561c	1562c	1569c	1582c	1584c	1724d	1756c	1762c	
	1770c	1771c	1773c	1776c	1785c	1792c	1794c	1795c	2005d	2041c	2047c	2058c	2059c	2061c	
LIFDAM(lifmodule)	2065c	2079c	2081c	2105c	2108c										
M8KSYS(ci)	583d	594c	595c	598c	625c	678d	748c	817c	836c	874c	879c	880c			
MOUSE(mouse)	875d	882c	885c	908c	1033d	1165d	1201c	1202c	1203c						
	75d	111c	119c	130d	134c	137c	162c	163c	169c	170c					
SRMDAM(srmdammodule)	465d	507d	518c	527c	535c	551c	573d	581c	586c	591c	596c	635c	668d	672c	
	673c	675c													
dontcare	54d	254c	261c												
doopendirectory															
LIFDAM(lifmodule)	561d	1153c													
doopenpack															
SRMDAM(srmdammodule)	900d	980c	1159c	1164c	1424c										
dooropen															
F9885(f9885dvr)	48d	219c													
dooropened															
F9885(f9885dvr)	62d	134c													
dooverwritefile															
LIFDAM(lifmodule)	1083d	1129c													
doprefix															
CTABLE	***	1234c													
INIT(fs)	***	760d													
M8KSYS(ci)		678c	679c	849c											
doprompt															
ETU	709d	768c													
dopurge															
LIFDAM(lifmodule)	668d	1148c													
do															
CS80(cs80)	***	220d													
doead	***	790d													
INIT(fs)	***	58d													
doreconfigure	***														
HPHIL(hphil)															
dorecover															
SRM_ORV(srm)	1356d	1395c	1473c												
dorese															
MOUSE(mouse)	67d	175c													
dosetsysdate															
CLOCK(clock)	100d	168c													
dosetsysime															
CLOCK(clock)	114d	189c													
dosysdate															
CLOCK(clock)	49d	166c													
dosysime															
CLOCK(clock)	73d	167c													
dotransfer															
ETU	834d	1249c													
double															
KERNEL(general_0)	968d	979c	1023c	1026c	1092c	1098c									
doupdatecursor															
CRT(crt)	269d	289c	311c	494c	567c	650c	653c	701c							
GCRT(crtb)	196d	220c	362c	513c											
down															
A804XDVR(a804xdvr)	***	49d													
CRT(crt)	***	97d	143d												
GCRT(crtb)	***	76d													
SYSDEVS(sysdevs)	***	93d													
downchar															
CRT(crt)	***	468c													
GCRT(crtb)	***	339c													
INIT(misc)	***	165d													
do															
LIFDAM(lifmodule)	115d	117c	113c												
dp															
GLE_RGL(gle_ras_out)	8162d	8206c													
dpt1															
GLE_RGL(gle_ras_out)	8149d	8150d	8162d	8163d	8164d	8165d	8166d	8167d	8168d	8169d	8170d	8171d	8172d	8173d	
	8174d	8175d	8176d	8177d	8178d										
dpt2															
GLE_RGL(gle_ras_out)	8150d	8162d													
dr															
LIFDAM(lifmodule)	585d	587c													
draw															
DGL_POLY(dgl_poly)	***	20825c													
GLE_GEN(gle_gen)	***	2066c													
GLE_HPGL(gle_hppl_out)	***	7528c													
GLE_RGL(gle_ras_out)	***	8397c													
GLE_SPARK(gle_smark)	***	5165c													
GLE_TYPES(gle_types)	***	1037d													
LIB(dgl_lib)		20141c	20608c												
draw_pg															
DGL_POLY(dgl_poly)	20317d	20902c													

11-47

II-48

[illegible]

eb54	CS80(cs80)	188d	315d																	
	CS80(cs80dsr)	725c																		
	TAPEEKUP(cs80tbdv)	607c	839c																	
eb56	CS80(cs80)	190d	316d																	
	CS80(cs80dsr)	727c																		
	TAPEEKUP(cs80tbdv)	609c	841c																	
eb60	CS80(cs80)	194d	319d																	
	CS80(cs80dsr)	731c																		
	TAPEEKUP(cs80tbdv)	613c	845c																	
eb62	CS80(cs80)	196d	320d																	
	CS80(cs80dsr)	733c																		
	TAPEEKUP(cs80tbdv)	615c	847c																	
eb63	CS80(cs80)	197d	321d																	
	CS80(cs80dsr)	657c	734c																	
	TAPEEKUP(cs80tbdv)	537c	616c	765c	848c															
eb_scar	CS80(cs80dsr)	641d	657c	658c	661c	662c	664c													
	TAPEEKUP(cs80tbdv)	521d	537c	538c	541c	542c	544c	701d	765c	766c	769c	770c	772c							
ebadars	EDRIVER(edriver)	34d	116c																	
ebank	EDRIVER(edriver)	33d	319c																	
ebate	EDRIVER(edriver)	47d	334d	340c																
	ETU	591c	1071c																	
ecode	EDRIVER(edriver)	105d	107c																	
ecfail	EDRIVER(edriver)	34d	291c																	
	ETU	582c	595c																	
echeck	EDRIVER(edriver)	33d	320c																	
	ETU	572c																		
echo	DGL_HPGLI(dgl_hpqli)	18017d	18031c	18036d	18049c	18051c	18054c	18061c	18069c	18087c	18105c	18112c	18114c							
	DGL_KNOB(dgl_knob)	18024d	18032c	18036d	18050c	18052c	18055c	18062c	18064c	18070c	18119c	18133c	18151c	18158c	18160c					
	DGL_VARS(dgl_vars)	1141d	1144d																	
	KEYS(keys)	456c	467c																	
	LIB(dgl_lib)	20076d	20080d	20934d	20943c	20951d	20996c	21017c												
	PRINTER(prtdvr)	157c	181c	184c																
	SYSDEV(Sysdevs)	238d	528c	533c	542c	546c	560c	564c	569c	638c										
echo_cursor	LIB(dgl_lib)	20172d	20232c	20234c																
echo_gcb	DGL_KNOB(dgl_knob)	18047d	18068c	18091c	18120c	18125c	18152c	18155c												
echo_mult	GLE_KNOB(gle_knob_in)	18017d	18171c	18189c																
echo_rate	GLE_KNOB(gle_knob_in)	18016d	18137c	18138c	18139c	18140c	18171c	18270c												
echoerror	DGL_HPGLI(dgl_hpqli)	18044d	18053c	18057c	18119c															
	DGL_KNOB(dgl_knob)	18044d	18054c	18058c	18165c															
ecode	EDRIVER(edriver)	79d	107c	128c	313d	322c	329d	331c	337d	340c	346d	348c								
	ETU	639d	641c	642c																
	INIT(ldr)	2408d	2489c	2491c	2492c	2504c														
	INITLOAD(bootdamodule)	616d	620c	621c	622c															
ect_tyr	INITLOAD(mini)	337d	341d																	
ecwrite	EDRIVER(edriver)	33d	318c																	
	ETU	578c	593c																	
edefs	INITLOAD(loader)	1067d	1127c	1133c																
edge	DGL_HPGLI(dgl_hpqli)	17227d	17228d	17229d	17230d	17231d	17232d	17233d	17234d	17235d	17236d	17237d	17238d	17239d	17240d					
	DGL_INQ(dgl_inq)	17241d	17242d																	
	DGL_FOLY(dgl_poly)	6078c																		
	DGL_FAS(dgl_faster)	20159c	20186c																	
	DGL_VARS(dgl_vars)	17562c	17563d	17564d	17565d	17566d	17567d	17568d	17569d	17570d	17571d	17572d	17573d	17574d	17575d					
	DGL_VARS(dgl_vars)	1035d																		
edge_a	DGL_FOLY(dgl_poly)	20513d	20532c																	
edge_b	DGL_FOLY(dgl_poly)	20514d	20526c																	
edge_index	DGL_FOLY(dgl_poly)	20390d	20484c	20485c	20503c	20504c	20536c	20539c	20542c	20673c	20680c	20684c	20726c	20736c						
edge_polygo	DGL_FOLY(dgl_poly)	20192d	21060c	21105c	21136c	21167c														
edge_vertex	DGL_FOLY(dgl_poly)	20388d	20484c	20485c	20536c	20673c	20680c	20684c												
edheader	ETU	865d	1075c	1076c	1077c	1078c	1079c	1080c	1081c	1082c										
edi_clr	DI_DRV(extdi)	168d																		
	DI_DRV(init_discint)	232c																		
edi_end	DI_DRV(extdi)	172d																		
	DI_DRV(init_discint)	227c																		
edi_init	DI_DRV(extdi)	155d																		
	DI_DRV(init_discint)	219c																		
edi_isr	DI_DRV(extdi)	156d																		
	DI_DRV(init_discint)	220c																		
edi_ppoll	DI_DRV(extdi)	167d																		
	DI_DRV(init_discint)	230c																		
edi_rdt	DI_DRV(extdi)	157d																		
	DI_DRV(init_discint)	221c																		
edi_rds	DI_DRV(extdi)	161d																		
	DI_DRV(init_discint)	225c																		
edi_rdw	DI_DRV(extdi)	159d																		
	DI_DRV(init_discint)	223c																		
edi_send	DI_DRV(extdi)	166d																		
	DI_DRV(init_discint)	229c																		
edi_set	DI_DRV(extdi)	169d																		
	DI_DRV(init_discint)	231c																		
edi_test	DI_DRV(extdi)	170d																		
	DI_DRV(init_discint)	233c																		
edi_ifr	DI_DRV(extdi)	165d																		
	DI_DRV(init_discint)	228c																		
edi_wtb	DI_DRV(extdi)	158d																		
	DI_DRV(init_discint)	222c																		


```

edi_wtc
DI_DRV(extdi)      *** 163d
DI_DRV(init_discint) *** 228c
edi_wtw
DI_DRV(extdi)      *** 160d
DI_DRV(init_discint) *** 224c
edit
M68KSYS(c1)        661d 716c 717c 718c 720c 721c 722c
editor
M68KSYS(c1)        39d 720c 1012c 1118c
edriver
EDRIVER(edriver)   *** 23d
ETU                *** 26d
efttable
ASCII              *** 321c
ETU                320c 903c
INIT(misc)         707c 710c 715c 718c 723c 727c
INITLOAD           1706c 1707c 1708c
INITLOAD(bootdammodule) 670c 709c
INITLOAD(sysglobals) *** 264d
LIFDAM(lifmodule)  421c 543c 544c 960c 1004c
SRHDAM(srdammodule) *** 92c
UCSD_AM(ucsd_am)   *** 272c
UCSD_DAM(ucsdmodule) 212c 291c 546c
efttableptrtype
INITLOAD(sysglobals) 175d 264d
efttabletype
INITLOAD(sysglobals) 170d 175d
eg_clr
G_DRV(extg)        *** 183d
G_DRV(init_gpio)   *** 247c
eg_init
G_DRV(extg)        *** 172d
G_DRV(init_gpio)   *** 237c
eg_isr
G_DRV(extg)        *** 173d
G_DRV(init_gpio)   *** 238c
eg_rdb
G_DRV(extg)        *** 174d
G_DRV(init_gpio)   *** 239c
eg_rds
G_DRV(extg)        *** 178d
G_DRV(init_gpio)   *** 243c
eg_rdw
G_DRV(extg)        *** 176d
G_DRV(init_gpio)   *** 241c
eg_set
G_DRV(extg)        *** 184d
G_DRV(init_gpio)   *** 246c
eg_test
G_DRV(extg)        *** 185d
G_DRV(init_gpio)   *** 248c
eg_tfr
G_DRV(extg)        *** 182d
G_DRV(init_gpio)   *** 245c
eg_wtb
G_DRV(extg)        *** 175d
G_DRV(init_gpio)   *** 240c
eg_wtc
G_DRV(extg)        *** 180d
G_DRV(init_gpio)   *** 244c
eg_wtw
G_DRV(extg)        *** 177d
G_DRV(init_gpio)   *** 242c

```

```

egetinfo
EDRIVER(edriver)   *** 42d 325d 331c
ETU                *** 660c
eglobl
INITLOAD(loader)  1070d 1127c 1135c
M68KSYS(c1)        180c 192c
eh_clr
H_DRV(exth)        *** 183d
H_DRV(init_hpib)   *** 248c
eh_end
H_DRV(exth)        *** 187d
H_DRV(init_hpib)   *** 243c
eh_init
H_DRV(exth)        *** 170d
H_DRV(init_hpib)   *** 235c
eh_isr
H_DRV(exth)        *** 171d
H_DRV(init_hpib)   *** 236c
eh_ppoll
H_DRV(exth)        *** 182d
H_DRV(init_hpib)   *** 246c
eh_rdb
H_DRV(exth)        *** 172d
H_DRV(init_hpib)   *** 237c
eh_rds
H_DRV(exth)        *** 176d
H_DRV(init_hpib)   *** 241c
eh_rdw
H_DRV(exth)        *** 174d
H_DRV(init_hpib)   *** 239c
eh_send
H_DRV(exth)        *** 181d
H_DRV(init_hpib)   *** 245c
eh_set
H_DRV(exth)        *** 184d
H_DRV(init_hpib)   *** 247c
eh_test
H_DRV(exth)        *** 185d
H_DRV(init_hpib)   *** 249c
eh_tfr
H_DRV(exth)        *** 180d
H_DRV(init_hpib)   *** 244c
eh_wtb
H_DRV(exth)        *** 173d
H_DRV(init_hpib)   *** 238c
eh_wtc
H_DRV(exth)        *** 178d
H_DRV(init_hpib)   *** 242c
eh_wtw
H_DRV(exth)        *** 175d
H_DRV(init_hpib)   *** 240c
eheap
INITLOAD(loader)  1066d 1126c 1132c
M68KSYS(c1)        182c 807c
eight_diget_epsilon
DGL_VARS(dgl_vars) *** 1150s
LIB(dgl_lib)       20519c 20520c 20915c 20917c
einit
EDRIVER(edriver)   *** 45d 343d 348c
ETU                *** 641c
eirbyte
KERNEL(generator_0) 1207c 1265c
KERNEL(decclarations) 499d 513d 527d

```


[illegible]

epsta1	29d	114c	116c	158c	170c	171c	190c										
EDRIVER(edriver	661c	672c	673c	689c	730c	731c	767c	788c	798c	849c	974c	976c	992c				
ETU																	
EDRIVER(edriver	70d	96d															
eptr																	
EDRIVER(edriver)	61d	70d															
epwrite																	
EDRIVER(edriver)	S2d	203c	317c														
erase																	
GLE_RGL(gle_ras_out)	***	8078d															
erase_mode	***	1203d															
DGL_VARS(dgl_var)																	
erase_support																	
GLE_HPGL(gle_hppl_out)	***	7699c															
GLE_TYPES(gle_types)	***	1094d															
erasecol																	
CRT(crt)	***	123d															
GCRT(crtb)	***	56d															
SYSDEVS(sysdevs)	***	77d															
eraseeos																	
CRT(crt)	***	124d															
GCRT(crtb)	***	57d															
SYSDEVS(sysdevs)	***	78d															
eread																	
EDRIVER(edriver)	33d	316c															
err																	
LIBRARIAN	3616d	3663c															
err_aspect																	
DGL_VARS(dgl_var)	***	1226d															
LIB(dgl_lib)	***	20356c															
err_bad_parms																	
DGL_INQ(dgl_inq)	6047c	6073c															
DGL_POLY(dgl_poly)	20181c	20936c															
DGL_VARS(dgl_var)	***	1227d															
LIB(dgl_lib)	20295c	20330c	20390c	20404c	20423c	20799c											
err_dis_int																	
DGL_VARS(dgl_var)	***	1223d															
GEN(dgl_gen)	***	3086c															
LIB(dgl_lib)	20582c	20612c	20641c	20672c	20731c												
err_dispec_loc																	
DGL_VARS(dgl_var)	***	1230d															
err_echo_dis_int																	
DGL_HPGLI(dgl_hppl)	***	18119c															
DGL_KNOB(dgl_knob)	***	18165c															
DGL_VARS(dgl_var)	***	1225d															

[illegible]

II-59

fbkls ze																
ASCII(asciimodule)	***	10d														
INIT(misc)		547c	559c	563c	567c	589c	592c	595c	626c	627c	629c	631c	636c	639c	646c	
		649c	659c	663c	664c	667c	668c									
INITLOAD	***	1697c														
INITLOAD(loader)		813d	839d	1120c												
INITLOAD(sysglobals)		21d	129d													
LIBRARIAN		133c	139c													
M68KSYS(c1)	***	816c														
UCSD_AM(ucsd_am)		43d	62c	63c	75c	76c	78c	79c	82c	84c	92c	94c	109c	119c	139c	
		142c	144c	162c	165c	184c	205c	221c	241c	247c						
UCSD_DAM(ucsdmodule)		39d	100c	162c	163c	218c	335c	365c	368c	370c	371c	416c	417c	418c	419c	
		483c	489c	495c	501c	503c	504c	517c	518c	519c	521c	547c	549c	550c	582c	
		584c	604c	606c												
		65c	86c													
UNITID(uio)																
fblockio																
INIT(fs)	***	789d														
fbdp																
INIT(ldr)	***	2378c														
INITLOAD(loader)		912d	1141c	1390c	1391c	1437c	1438c	1605c	1610c							
LIBRARIAN		1296c	1299c	1393c	2258c	2299c	2771c	2817c	2867c	3018c						
fbufchanged																
ASCII(asciimodule)		39c	48c	69c	94c	117c	136c	253c	265c	271c	298c					
ETU	***	324c														
INIT(misc)	***	544c	550c	632c	650c	665c	678c									
INITLOAD(sysglobals)	***	89d														
LIFDAM(lifmodule)	***	298c														
UCSD_AM(ucsd_am)		64c	100c	140c												
UCSD_DAM(ucsdmodule)	***	84c														
fbuffer																
ASCII(asciimodule)		37c	50c	74c	94c	115c	116c	159c	203c	223c	251c					
INIT(misc)	***	263c	548c	564c	567c	598c	600c	629c	631c	646c	649c	659c	664c	675c	677c	
INITLOAD(sysglobals)	***	129d														
UCSD_AM(ucsd_am)		63c	76c	80c	83c	84c	92c	98c	117c	165c	203c	234c	245c			
UCSD_DAM(ucsdmodule)		500c	501c	503c	504c											
fbuffered																
ETU		321c	902c													
INIT(misc)	***	471c														
INITLOAD(bootdamodule)	***	713c														
INITLOAD(loader)	***	1607c														
INITLOAD(sysglobals)	***	80d														
LIFDAM(lifmodule)		298c	920c													
SRMDAM(srmdamodule)	***	948c														
UCSD_DAM(ucsdmodule)		84c	164c													
fbufferref																
INIT(fs)	***	776d														
fbufvalld																
INITLOAD(sysglobals)	***	85d														
LIFDAM(lifmodule)	***	283c														
UCSD_DAM(ucsdmodule)	***	78c														
fbusy																
AMIGD(amigodvr)	***	587c														
BUBBLES(bubble)		115c	116c													
CS80(cs80dvr)	***	1373c														
F9885(f9885dvr)	***	272c														
GLE_KNOB(gle_knob_in)	***	18068c														
INITLOAD(mini)	***	500c														
INITLOAD(sysglobals)	***	116d														
KEYS(keys)	***	83c														
M68KSYS(c1)		120c	140c	141c	382c											
UNITID(uio)		110c	111c													
fclose																
INIT(fs)	***	787d														
fcloseit																
INIT(fs)	***	766d														
fd																
F9885(f9885dvr)		53d	111d	115d												
fdate																
INITLOAD(bootdamodule)	***	563d														
LIFDAM(lifmodule)		64d	372c	548d	550c	551c	552c	555c	556c	557c	602c	617c	651c	652c	940c	
		941c	1008c	1093c	1094c											
fdirectory																
INIT(ldr)		2436c	2437c	2446c	2447c	2452c										
INITLOAD(loader)		1053d	1144c	1145c	1146c	1152c	1161c	1163c	1318c	1621c	1622c					
LIBRARIAN		1288c	1324c	1352c	1386c	2854c	2855c	2959c	2967c	3020c	3022c	3029c	3036c	3039c	3064c	
		3065c	3068c	3087c	3097c	3255c	3256c	3279c	3280c	3312c	3326c	3328c	3348c	3361c	3372c	
		3374c	3428c	3433c	3438c	3452c	3465c	3471c	3490c	3586c	3605c	3623c	3631c	3632c	3642c	
		3652c	3735c													
SEGMENTER(asm)		194c	195c													
fdp																
INITLOAD(loader)		910d	1144c	1152c	1159c	1161c										
LIBRARIAN		2854c	2859c	2888c	3106c	3112c	3153c									
feft																
ETU		320c	903c													
INITLOAD(sysglobals)	***	77d														
LIFDAM(lifmodule)		937c	959c	1110c												
SRMDAM(srmdamodule)		934c	938c	939c	940c	1045c	1047c	1090c	1387c	1418c	1452c					
UCSD_DAM(ucsdmodule)		212c	291c													
feof																
ETU		327c	468c	472c	481c	1034c	1045c									
INIT(fs)	***	775d														
INITLOAD(sysglobals)	***	87d														
LIFDAM(lifmodule)	***	290c														
UCSD_DAM(ucsdmodule)	***	79c														
feoln																
ASCII(asciimodule)		207c	208c													
ETU		327c	459c	461c	1034c											
INIT(fs)	***	780d														
INIT(misc)	***	525c	806c	684c	685c											
INITLOAD(sysglobals)	***	86d														
KEYS(keys)	***	127c														
M68KSYS(c1)		348c	351c													
UCSD_AM(ucsd_am)		152c	230c	253c	254c											
feot																
AMIGD(amigodvr)	***	600c														
CRIB(crib)		448c	496c													
CS80(cs80dvr)	***	1340c														
DISC4PIB(bkgnd)		42d	139c													
F9885(f9885dvr)	***	298c														
GCRIB(crib)		316c	364c													
GLE_KNOB(gle_knob_in)	***	18040c														
INIT_LOAD(mini)	***	528c														
INIT_LOAD(sysglobals)	***	118d														
KEYS(keys)	***	136c														
UNITID(uio)	***	47c														
fetchbuffer																
INIT(misc)		554d	593c	644c	658c	673c										
fetchdir																
UCSD_DAM(ucsdmodule)		70d	121c	128c	463c	654c	655c	660c								
fetchvolume																
UCSD_DAM(ucsdmodule)		124d	127c	133c	211c	269c	331c	479c	512c	533c	578c					
fextra																
INIT_LOAD(sysglobals)	***	123d														
fextra2																
INIT_LOAD(sysglobals)	***	124d														

II-60

fileinfo	INITLOAD(sysglobals)	20d	43d												
	SRMDAM(srmdammodule)	***	1682c												
fidptr	SRMDAM(srmdammodule)	1173d	1193c	1194c	1195c										
field	H88KSYS(c1)	209d	225c												
fieldwidth	MUII	228d	261c												
TAPEBACKUP		1115d	1148c												
file_code	SRMDAM(srmdammodule)	803c	604c	924c	934c	935c	1109c								
	SRM_DRV(srm)	278d	495d	863d	1762c										
file_header_type	SRM_DRV(srm)	266d	377d	400d	423d	494d	520d	521d	538d	539d	647d	696d	1512d		
file_id	SRMDAM(srmdammodule)	819c	848c	927c	931c	1101c									
	SRM_DRV(srm)	1856c	1893c	1954c	2010c	2035c	2055c	2096c							
	SRM_DRV(srm)	456d	572d	630d	659d	678d	712d	734d	753d	769d	802d	1690c			
file_id_1	SRM_DRV(srm)	***	1836c												
	SRM_DRV(srm)	***	555d												
file_id_2	SRM_DRV(srm)	***	1837c												
	SRM_DRV(srm)	***	556d												
file_id_type	SRMDAM(srmdammodule)	657d	658d	1338d											
	SRM_DRV(srm)	1784d	1789d	1828d	1827d	1846d	1882d	1906d	1942d	1969d	1986d	2023d	2046d	2082d	
	SRM_DRV(srm)	198d	268d	458d	474d	478d	555d	556d	572d	630d	659d	678d	712d	734d	753d
		789d	802d	1009d	1017d	1023d	1034d	1038d	1038d	1045d	1059d	1084d	1068d	1089d	1071d
		1075d	1081d	1086d	1093d	1096d	1102d	1106d	1109d	1514d	1551d	1589d	1627d	1682d	1703d
		1705d	1732d												
file_index	SRM_DRV(srm)	374d	1586c												
file_info	SRMDAM(srmdammodule)	422c	1103c	1445c	1485c										
	SRM_DRV(srm)	579d													
file_info_type	SRM_DRV(srm)	274d	386d	578d											
file_init	DGL_C_OUT(dgl_confg_out)	***	11443c												
	GLE_FILE(gle_file_io)	9017d	9049d												
file_set_timeout	DGL_C_OUT(dgl_confg_out)	***	11435c												
	GLE_FILE(gle_file_io)	9019d	9037d												
file_tcb	DGL_C_OUT(dgl_confg_out)	***	11035d												
	GLE_FILE(gle_file_io)	9009d	9010d												
file_tcb_ptr	DGL_C_OUT(dgl_confg_out)	11370d	11402d												
	GLE_FILE(gle_file_io)	9009d	9055c	9065c	9075c										
file_tcb_space	DGL_C_OUT(dgl_confg_out)	11035d	11431c												
file_len	MUII	412d	446d												
file_name	DGL_C_OUT(dgl_confg_out)	***	11439c												
	GLE_FILE(gle_file_io)	9011d	9057c												
	MUII	409d	443d												
	SRMDAM(srmdammodule)	127c	151c	156c	165c	431c	432c	602c	767c	770c	787c	857c	1004c	1191c	1195c
	SRM_DRV(srm)	1196c	290d												
		275d													
file_name_header	SRM_DRV(srm)	1922c	1983c												
	SRM_DRV(srm)	377d	400d	423d	494d	647d	696d	1573c	1611c	1644c	1756c				
file_name_header1	SRM_DRV(srm)	***	538d												
file_name_header2	SRM_DRV(srm)	***	538d												
file_name_set	SRMDAM(srmdammodule)	120d	463d	506d	572d	966d	994d	1178d	1181d	1217d	1220d	1249d	1334d		
	SRM_DRV(srm)	289d	926d												
file_password	SRM_DRV(srm)	458d	1692c												
file_set_timeout	DGL_C_OUT(dgl_confg_out)	***	11436c												
	GLE_FILE(gle_file_io)	9020d	9043d												
file_term	DGL_C_OUT(dgl_confg_out)	11382c	11453c												
	GLE_FILE(gle_file_io)	9021d	9072d												
file_type	MUII	410d	444d												
file_write	DGL_C_OUT(dgl_confg_out)	***	11433c												
	GLE_FILE(gle_file_io)	9018d	9062d												
fileblock	INITLOAD(loader)	1029d	1343c	1539c	1544c										
	LIBRARIAN	1296c	1299c	1394c	2632c	2633c	2771c	2817c							
filedirectory	INITLOAD(loader)	879d	1033d												
fileid_ptr	INITLOAD(loader)	879d	910d	1053d											
filelib	INITLOAD(loader)	1028d	1342c	1437c	1438c	1538c	1543c								
	LIBRARIAN	1296c	1299c	1393c	2258c	2299c	2771c	2817c							
fileid	AMIG0(amigodvr)	***	624c												
	BUBBLES(bubble)	***	77c	94c											
	CS80.cs80dvr)	***	1395c												
	EPROMS(eproms)	***	57c												
	F885(f885dvr)	***	286c												
	INIT misc)	***	288c	452c											
	INITLOAD(bootdammodule)	***	710c	711c											
	INITLOAD(mini)	***	514c												
	INITLOAD(sysglobals)	***	107d												
	L1FDH(l1fmodule)	***	248c	291c	408c	646c	915c								
	H88KSYS(c1)	***	738c	740c	956c										
	MUII(bminit)	***	77c												
	MUII	***	488c												
	SRMDAM(srmdammodule)	67c	75c	84c	114c	122c	131c								
	SRMDAM(srmdammodule)	332d	334c	335c	337c	340c	342c	931c	1279c	1281c	1285c	1319c	1347c	1349c	1352c
	UCSD_DAM(ucsdmodule)	1380c	1389c	1411c	1426c	1427c	1429c	1449c	1462c	1477c	1483c	1508c	1510c	1677c	1804c
		80c	162c	163c	330c	335c	502c	585c							
file_infopack	SRMDAM(srmdammodule)	***	421c	1102c	1462c	1483c									
	SRM_DRV(srm)	***	1845d												
	SRM_DRV(srm)	***	1070d												

[illegible]

[illegible]

[illegible]

II-66

fproof																
AMIG00(amigodvr)	***	610c														
ASCI00(asciimodule)		21c	78c	82c	132c											
BUBBLES(bubble)		74c	91c													
CS80(cs80dvr)	***	1392c														
EPROMS(eproms)	***	54c														
ETU	***	909c														
F9885(f9885dvr)	***	284c														
INIT(misc)		267c	269c	272c	454c	455c	457c	462c	466c	496c	499c	615c	618c			
INITLOAD(bootdammodule)		705c	712c	757c												
INITLOAD(misc)	***	512c														
INITLOAD(sysglobals)	***	93d														
LIFDAM(lifmodule)		248c	294c	295c	409c	593c	595c	606c	607c	916c	918c	1054c				
MINIT(bminit)	***	76c														
MUI	***	489c														
SRMDAM(srmdammodule)		481c	527c	537c	557c	586c	598c	640c	932c	1322c	1487c					
UCSD_AH(ucsd_am)		55c	58c													
UCSD_DAM(ucsdmodule)		81c	100c	101c	102c	163c	294c	295c	371c	496c	536c	563c	585c			
fpnydata																
MUI	***	475c														
fpos																
ASCII(asciimodule)		28c	60c	77c	80c	83c	93c	94c	95c	107c	112c	120c	131c	157c	158c	
		159c	160c	169c	200c	201c	202c	203c	204c	208c	221c	222c	223c	224c	228c	
ETU		241c	247c	249c	251c	252c	255c	266c	273c	274c	299c					
INIT(misc)		322c	436c	453c	479c	1035c										
INITLOAD(sysglobals)	***	491c	492c	496c	499c	501c	574c	587c	592c	594c	602c	610c	611c	613c	615c	
LIFDAM(lifmodule)		618c	622c	639c												
M68KSYS(c1)	***	91d														
SRMDAM(srmdammodule)	***	296c	593c	931c	1022c											
UCSD_AH(ucsd_am)		90c														
UCSD_DAM(ucsdmodule)		526c	585c	1056c	1057c	1318c	1413c	1414c	1415c	1450c						
		52c	53c	55c	58c	60c	62c	64c	94c	97c	116c	118c	139c	153c	161c	
		182c	165c	184c	202c	204c	208c	218c	221c	237c	240c	246c	252c	260c		
UCSD_DAM(ucsdmodule)		82c	217c	366c	368c	535c										
fposition																
INIT(fs)	***	773d														
fpurgeoldlink																
INITLOAD(sysglobals)	***	111d														
SRMDAM(srmdammodule)	***	1237c														
fput																
INIT(fs)	***	778d														
frac																
GEN(dgl_gen)		3169d	3175c	3177c	3178c											
fracstate																
MOREFSYS(mfs)		194d	244c	282c	322c	361c										
fractionadjust																
GLE_STEXT(gle_stext)		4021d	4023d	4024d	4047c	4048c	4051c	4052c								
framebuffer																
DGL_C_OUT(dgl_config_out)		11028d	11080d	11110c	11134c	11258c	11269c									
frameaddr																
GCRT(crtb)		159c	473c													
SYSDEVS(sysdevs)	***	132d														
fread																
INIT(fs)	***	783d														
freadable																
INIT(ldr)		2296c	2304c													
INITLOAD(sysglobals)	***	83d														
LIFDAM(lifmodule)		288c	315c	1127c												
M68KSYS(c1)	***	90c														
MOREFSYS(mfs)	***	520c														
UCSD_DAM(ucsdmodule)		77c	630c	675c												
freadpool																
INIT(fs)	***	813d														
freadbytes																
INIT(fs)	***	786d														
freadchar																
INIT(fs)	***	802d														
freadenum																
INIT(fs)	***	811d														
freadint																
INIT(fs)	***	804d														
freadin																
INIT(fs)	***	800d														
freadmode																
INITLOAD(sysglobals)	***	85d														
LIFDAM(lifmodule)	***	289c														
UCSD_DAM(ucsdmodule)	***	78c														
freadpaoc																
INIT(fs)	***	810d														
freadreal																
MOREFSYS(mfs)		18d	515d													
freadstr																
INIT(fs)	***	808d														
freadstrbool																
INIT(fs)	***	836d														
freadstrchar																
INIT(fs)	***	817d														
freadstrenum																
INIT(fs)	***	832d														
freadstrpaoc																
INIT(fs)	***	830d														
freadstrreal																
MOREFSYS(mfs)		21d	533d													
freadstr																
INIT(fs)	***	825d														
freadstrword																
INIT(fs)	***	820d														
freadword																
INIT(fs)	***	803d														
frecsize																
INITLOAD(sysglobals)	***	76d														
free_blocks																
SRM_DRV(srm)	***	345d														
freearray																
HEAPT(hpm)		75d	79d	123d												
freelist																
HEAPT(hpm)		79d	123d	145c	147c	148c	185c	186c	205c	206c	223c	226c	233c	236c	241c	
		252c														
french_kbd																
A804XDVR(a804xdvr)		191d	193d													
NONUSKBD1(non_us_kbd1)		77d	117c	186c												
NONUSKBD2		155c	216c													
SYSDEVS(sysdevs)	***	149d														
freipient																
ASCII(asciimodule)		105c	112c	115c	116c	120c	129c	168c	169c	170c	171c	184c	185c	187c	192c	
		196c	202c	204c	207c	218c	222c	224c	226c	239c	248c	252c	295c	305c		
ETU	***	322c														
INITLOAD(sysglobals)	***	96d														
LIFDAM(lifmodule)	***	297c														
UCSD_AH(ucsd_am)		125c	127c	128c	129c	130c	192c	196c	199c	201c	229c	230c	248c			
UCSD_DAM(ucsdmodule)	***	83c														
frequency																
A804XDVR(a804xdvr)		157d	160c													
SYSDEVS(sysdevs)		367d	416d	417c												

II-68

[illegible]

II-70

getextable							
INIT_LOAD(loader)	1281d	1345c					
getinfo							
EDRIVER(edriver)	122d	296c					
getinfobytes	478d	491c	528c	687c	780c	945c	974c
LIBKARIAN							
getinfostruction	488d	1195c	1211c	1239c			
LIBKARIAN							
getioerrmsg							
ETU	***	104c					
INIT(ldr)	***	2414c					
INIT(misc)		203d	300d	389c			
LIBKARIAN	***	3663c					
M8KSYS(ci)	***	682c					
MIU		26c	144c				
TAPEBKUP		990c	1074c				
TAPEBKUP(cs80tdvr)	***	746c					
getpac							
SRMDAM(srmdamodule)		664d	727c	770c	778c	787c	
getrealnumber							
MOREFSYS(mfs)		396d	528c	548c			
getrecsize							
ASCII(asciimodule)		164d	195c	285c			
getsdate							
LIFDAM(lifmodule)		324d	1149c				
getsize							
HEAPT(hpm)		99d	106c	165c	171c	244c	
getspace							
LIFDAM(lifmodule)		676d	931c				
gettirfo							
A804XDVR(a804xdvr)	***	314c					
SYSEVS(sysevs)		214d	480c				
getuntildelim							
SRMDAM(srmdamodule)		217d	250c	265c	275c		
getvalue							
M8KSYS(ci)		745d	768c	797c	800c	803c	846c
getvoimedate							
INIT_LOAD(sysglobals)	***	133d					
LIFDAM(lifmodule)	***	1149c					
M8KSYS(ci)	***	160c					
SRMDAM(srmdamodule)	***	1791c					
UCSD_DAM(ucsdmodule)	***	654c					
getvoimedate							
INIT(misc)	***	446c					
INIT_LOAD(bootdamodule)	***	692c					
INIT_LOAD(sysglobals)	***	133d					
LIFDAM(lifmodule)	***	1134c					
M8KSYS(ci)	***	616c					
MIU		249c	529c				
SRMDAM(srmdamodule)	***	1789c					
TAPEBKUP	***	1136c					
UCSD_DAM(ucsdmodule)	***	659c					
getxy							
CRT(crt)		293d	413c				
GCRT(crtb)		202d	283c				

gfiles							
CRT(crt)		198c	256c	259c			
C.HOOK		109c	113c				
DGL_RAS(dgl_raster)		17339c	17343c	17393c	17417c	17421c	
GCRT(crtb)		161c	185c	189c			
INIT		2523c	2524c	2525c	2528c	2545c	
INIT(ldr)		2290c	2291c	2304c	2305c	2306c	2308c 2309c
INIT_LOAD(sysglobals)	***	257d					
M8KSYS(ci)		117c	119c	547c	548c	567c	568c
SWVOL		22c	24c				
gheight							
CRT(crt)		218d	225d	248c			
gheightb							
CRT(crt)		218d	226d	242c			
gint list							
DGL_HPGL(dgl_hpgl)		17019d	17064d				
DGL_INQ(dgl_inq)		6013d	6089d				
DGL_POLY(dgl_poly)		20045d	20193d	20317d	20844d		
DGL_RAS(dgl_raster)		17360d	17429d				
DGL_VARS(dgl_vars)		1124d	1129d				
LIB(dgl_lib)		20059d	20066d	20753d	20771d		
LIB(dgl_lib)	***	1012d					
gTABLE(scanstuff)		1064d	1105c				
gle							
GLE_UTILS(gle_util)	***	21008d					
gle_aras_out							
GLE_RGL(gle_ras_out)	***	8004d					
gle_asclip							
GLE_HPGL(gle_hpgl_out)	***	7035d					
GLE_RGL(gle_ras_out)	***	8089d					
gle_astext							
GLE_HPGL(gle_hpgl_out)	***	7034d					
GLE_RGL(gle_ras_out)	***	8087d					
gle_autl							
DGL_RAS(dgl_raster)	***	17017d					
GLE_RGL(gle_ras_out)	***	8091d					
GLE_SHARK(gle_shark)	***	5014d					
gle_wait_blanking							
DGL_RAS(dgl_raster)	***	17459c					
GLE_GEN(gle_gen)		2041d	2255d				
LIB(dgl_lib)	***	20230c					
gle_buffer_mode							
GLE_GEN(gle_gen)		2033d	2219d				
LIB(dgl_lib)	***	20395c					
gle_byte							
GLE_TYPES(gle_types)	***	1006d					
gle_char6							
GLE_TYPES(gle_types)		1012d	1078d	1080d	1227d	1229d	
gle_char_size							
GLE_GEN(gle_gen)		2011d	2075d				
LIB(dgl_lib)	***	20836c					
gle_clib							
DGL_HPGL(dgl_hpgl)	***	17295c					
DGL_RAS(dgl_raster)		17528c	17614c				
GLE_GEN(gle_gen)	***	2017d	2093d				
LIB(dgl_lib)	***	20551c					
gle_clib_limits							
GLE_GEN(gle_gen)	***	3545c					
GLE_GEN(gle_gen)		2018d	2117d				
gle_copy_to_string							
GLE_FILE(gle_file_io)	***	9057c					
GLE_UTILS(gle_util)		21010d	21106d				

gle input_echo		
DGL_HPGLI(dgl_hpqli)	18032c	18115c
DGL_KNOB(dgl_knob)	18033c	18161c
GLE_GENI(gle_geni)	3011d	3049d
gle input_escape1		
GLE_GENI(gle_geni)	3012d	3055d
gle input_escape0		
GLE_GENI(gle_geni)	3013d	3061d
gle input_term		
GLE_GENI(gle_geni)	3010d	3042d
LIE(dgl_lib)	***	21086c
gle ishift		
GLE_SMARK(gle_smark)	5158c	5159c
gle knob_echo_gcb		
DGL_KNOB(dgl_knob)	18068c	18072c 18077c
DGL_VARS(dgl_vars)	***	1335s
LIE(dgl_lib)	21395c	21397c
gle knob_echo_gcb_space		
DGL_VARS(dgl_vars)	***	1341d
LIE(dgl_lib)	***	21395c
gle knob_in		
DGL_C_IN(dgl_cfg_in)	***	12013d
GLE_KNOB(gle_knob_in)	***	18002d
gle linestyle		
DGL_HPGLI(dgl_hpqli)	***	17201c
DGL_RAS(dgl_raster)	***	17061c
GLE_GENI(gle_geni)	2031d	2207d
LIE(dgl_lib)	20985c	21006c
gle linewidth		
GLE_GENI(gle_geni)	2032d	2213d
LIE(dgl_lib)	20443c	20988c 21009c
gle marker		
GLE_GENI(gle_geni)	2015d	2105d
LIE(dgl_lib)	***	20746c
gle marker_size		
GLE_GENI(gle_geni)	2023d	2157d
LIE(dgl_lib)	***	21185c
gle match		
DGL_HPGLI(dgl_hpqli)	17035c	17036c 17037c 17038c 17039c 17040c 17123c 17137c 17138c 17146c 17147c 17148c 17149c 17150c
GLE_HPGLI(gle_hpqli_out)	17151c	17157c 17158c
	7232c	7233c 7234c 7253c 7254c 7271c 7272c 7280c 7281c 7289c 7297c 7616c 7619c 7620c
	7627c	7634c 7635c 7636c 7643c 7644c 7645c 7646c 7647c 7648c 7649c 7650c 7651c 7652c
	7660c	7661c 7662c 7663c 7666c 7667c
GLE_HPGLI(gle_hpqli_in)	18140c	18141c 18142c 18158c 18159c 18167c 18168c 18176c 18184c 18192c
GLE_UTLS(gle_utls)	21011d	21118d 21131c 21133c
gle move		
DGL_POLY(dgl_poly)	20229c	20238c
GLE_GENI(gle_geni)	2008d	2057d
LIE(dgl_lib)	***	20242c
gle output_escape1		
DGL_HPGLI(dgl_hpqli)	***	17050c
GLE_GENI(gle_geni)	2039d	2051d
gle output_escape0		
DGL_HPGLI(dgl_hpqli)	17047c	17179c
GLE_GENI(gle_geni)	2040d	2045d
gle polygon		
DGL_POLY(dgl_poly)	21053c	21098c
GLE_GENI(gle_geni)	2027d	2183d
gle ras_out		
DGL_C_OUT(dgl_cfg_out)	***	11012d
DGL_INQ(dgl_inq)	***	6030d
DGL_RAS(dgl_raster)	***	17018d
GLE_RGL(gle_ras_out)	***	8002d
gle read_integer		
DGL_C_IN(dgl_cfg_in)	12049c	12097c
DGL_C_OUT(dgl_cfg_out)	11279c	11419c
GLE_HPGLI(gle_hpqli_out)	7182c	7184c
GLE_HPGLI(gle_hpqli_in)	18108c	18110c 7188c 7188c 7242c 7244c 7246c 7248c
GLE_HPGLI(gle_hpqli_io)	***	18108c 18110c 18112c 18114c 18148c 18150c 18152c 18154c 18254c 18255c 18265c 18301c 18303c 18305c
GLE_UTLS(gle_utls)	0493c	
GLE_UTLS(gle_utls)	21008d	21069d 21103c
gle sample		
DGL_HPGLI(dgl_hpqli)	18028c	18079c
DGL_KNOB(dgl_knob)	***	18110c
GLE_GENI(gle_geni)	3014d	3067d
gle sclip		
GLE_HPGLI(gle_hpqli_out)	***	7036d
GLE_RGL(gle_ras_out)	***	8088d
GLE_SCLIP(gle_sclip)	***	6002d
gle set marker		
GLE_GENI(gle_geni)	2022d	2151d
LIE(dgl_lib)	***	20745c
gle sh		
GLE_UTLS(gle_utls)	***	21069d
gle shortint		
DGL_C_IN(dgl_cfg_in)	12043d	12089d
DGL_C_OUT(dgl_cfg_out)	11158d	11406d
DGL_POLY(dgl_poly)	20552d	20553d 20554d
DGL_RAS(dgl_raster)	17203d	17250d
GLE_FILE(gle_file_io)	9012d	9014d
GLE_HPGLI(gle_hpqli_out)	7066d	7069d
GLE_HPGLI(gle_hpqli_in)	18039c	18042d 18054d 18056d 18079d 18096d 18097d 18134d 18135d 18226d 18227d 18230d 18291d
GLE_HPGLI(gle_hpqli_io)	10013d	10486d
GLE_KNOB(gle_knob_in)	18011d	18016d 18017d
GLE_RGL(gle_ras_out)	8016d	8017d 8018d 8030d 8031d 8033d 8038d 8039d 8040d 8041d 8042d 8043d 8044d 8045d
	8047d	8048d 8049d 8050d 8051d 8052d 8053d 8054d 8060d 8061d 8065d 8066d 8094d 8095d
	8247d	8293d 8312d 8314d 8320d 8377d
	5096d	5097d 5098d 5099d 5100d 5101d
GLE_SMARK(gle_smark)	4008d	
GLE_TEXT(gle_text)	***	1007d
GLE_TYPES(gle_types)	1099d	1074d 1075d 1076d 1079d 1081d 1091d 1092d 1093d 1094d 1095d 1096d 1097d 1098d
	1146d	1100d 1103d 1104d 1105d 1106s 1115d 1119d 1135d 1139d 1141d 1142d 1143d 1144d
	1224d	1148d 1149d 1151d 1152d 1153d 1154d 1166d 1167d 1168d 1169d 1171d 1187d
	1225d	1228d 1230d 1249d 1268d 1269d 1270s 1274d 1275d 1276d
GLE_UTLS(gle_utls)	21008d	21009d 21010d 21011d 21012d 21013d 21014d 21019d 21028d 21037d 21038d 21045d 21046d 21047d
	21069d	21074d 21076d 21077d 21078d 21106d 21110d 21118d 21119d 21123d
gle shortint_max		
GLE_KNOB(gle_knob_in)	18174c	18175c
GLE_UTLS(gle_utls)	21014d	21028d 21033c 21034c
gle shortint_min		
GLE_KNOB(gle_knob_in)	18174c	18175c
GLE_UTLS(gle_utls)	21013d	21019d 21024c 21025c
gle smark		
GLE_HPGLI(gle_hpqli_out)	***	7037d
GLE_RGL(gle_ras_out)	***	8090d
GLE_SMARK(gle_smark)	***	5002d
gle soft_char_size		
GLE_HPGLI(gle_hpqli_out)	***	7531c
GLE_RGL(gle_ras_out)	***	8401c
GLE_TEXT(gle_text)	4010d	4058d
gle soft_clip_draw		
GLE_HPGLI(gle_hpqli_out)	***	7528c
gle soft_clip_limits		
GLE_HPGLI(gle_hpqli_out)	7532c	7684c
GLE_RGL(gle_ras_out)	8400c	8511c
GLE_SCLIP(gle_sclip)	6008d	6012d
gle soft_clip_move		
GLE_HPGLI(gle_hpqli_out)	***	7527c

gle_soft_marker			
GLE_HPGL(gle_hppl_out)	***	7537c	
GLE_RGL(gle_ras_out)	***	8405c	
GLE_SMARK(gle_smark)		5008d	5016d
gle_soft_marker_size			
GLE_HPGL(gle_hppl_out)	***	7538c	
GLE_RGL(gle_ras_out)	***	8406c	
GLE_SMARK(gle_smark)		5010d	5185d
gle_soft_set_marker			
GLE_HPGL(gle_hppl_out)	***	7539c	
GLE_RGL(gle_ras_out)	***	8407c	
GLE_SMARK(gle_smark)		5009d	5178d
gle_soft_text			
GLE_HPGL(gle_hppl_out)	***	7530c	
GLE_RGL(gle_ras_out)	***	8399c	
gle_soft_text_dir			
GLE_HPGL(gle_hppl_out)	***	7535c	
GLE_RGL(gle_ras_out)	***	8403c	
GLE_STEXT(gle_stext)		4012d	4082d
gle_soft_text_just			
GLE_HPGL(gle_hppl_out)	***	7536c	
GLE_RGL(gle_ras_out)	***	8404c	
GLE_STEXT(gle_stext)		4013d	4094d
gle_soft_text_spacing			
GLE_HPGL(gle_hppl_out)	***	7533c	
GLE_RGL(gle_ras_out)	***	8402c	
GLE_STEXT(gle_stext)		4011d	4070d
gle_start_digitize			
DGL_HPGLI(dgl_hppli)	***	18064c	
DGL_KNOB(dgl_knob)	***	18104c	
GLE_GENI(gle_geni)		3015d	3073d
gle_stext			
GLE_HPGL(gle_hppl_out)	***	7033d	
GLE_RGL(gle_ras_out)	***	8086d	
GLE_STEXT(gle_stext)	***	4002d	
gle_term			
GLE_GEN(gle_gen)		2025d	2170d
LIB(dgl_lib)	***	21053c	
gle_text			
GLE_GEN(gle_gen)		2010d	2069d
LIB(dgl_lib)	***	20728c	
gle_text_dir			
GLE_GEN(gle_gen)		2013d	2087d
LIB(dgl_lib)	***	20811c	
gle_text_just			
GLE_GEN(gle_gen)		2014d	2099d
gle_text_spacing			
GLE_GEN(gle_gen)		2012d	2081d
LIB(dgl_lib)	***	20840c	
gle_text_xform			
GLE_HPGL(gle_hppl_out)	***	7545c	
GLE_RGL(gle_ras_out)	***	8414c	
GLE_STEXT(gle_stext)		4014d	4018d

gle_types			
DGL_C_IN(dgl_conf_in)	***	12004d	
DGL_C_OUT(dgl_conf_out)	***	11004d	
DGL_HPGLI(dgl_hppli)	***	18012d	
DGL_KNOB(dgl_knob)	***	18012d	
DGL_POLY(dgl_poly)	***	20036d	
DGL_RAS(dgl_raster)	***	17015d	
DGL_TOOLS(dgl_tools)	***	20011d	
DGL_VARS(dgl_vars)	***	52c	
GEN(dgl_gen)	***	3056d	
GLE_FILE(gle_file_io)	***	9004d	
GLE_GEN(gle_gen)	***	2004d	
GLE_GENI(gle_geni)	***	3004d	
GLE_HPGL(gle_hppl_out)	***	7004d	
GLE_HPGLI(gle_hppli_in)	***	18004d	
GLE_HPIB(gle_hpib_io)	***	10004d	
GLE_KNOB(gle_knob_in)	***	18004d	
GLE_RGL(gle_ras_out)	***	8004d	
GLE_SCLIP(gle_sclip)	***	6004d	
GLE_SMARK(gle_smark)	***	5004d	
GLE_STEXT(gle_stext)	***	4004d	
GLE_TYPES(gle_types)	***	1002d	
GLE_UTLS(gle_utls)	***	21004d	
LIB(dgl_lib)	***	20104d	
gle_utls			
DGL_C_IN(dgl_conf_in)	***	12014d	
DGL_C_OUT(dgl_conf_out)	***	11015d	
DGL_HPGL(dgl_hppl)	***	17013d	
GLE_FILE(gle_file_io)	***	9025d	
GLE_HPGL(gle_hppl_out)	***	7038d	
GLE_HPIB(gle_hpib_io)	***	10036d	
GLE_KNOB(gle_knob_in)	***	18028d	
GLE_UTLS(gle_utls)	***	21002d	
gle_write_integer			
GLE_HPGL(gle_hppl_out)	***	7088c	
GLE_HPGLI(gle_hppli_in)	***	18061c	
GLE_UTLS(gle_utls)		21009d	21037d
gload			
GLE_GEN(gle_gen)	***	2234c	
GLE_HPGL(gle_hppl_out)	***	7554c	
GLE_TYPES(gle_types)	***	1043d	
global			
INITLOAD(loader)	***	858d	
LIBRARIAN		181c	1586c 1617c
SEGMENTER(asm)		96d	99c
SEGMENTER(segmenter)	***	38d	
globalbase			
INITLOAD(loader)		953d	1412c
LIBRARIAN		370c	1685c 1971c
globaldelta			
INITLOAD(loader)		1027d	1412c
LIBRARIAN		1495c	1496c 1685c
globalsize			
INITLOAD(loader)		952d	1379c 1413c
LIBRARIAN		371c	1686c 1970c 2617c
globalsize			
INITLOAD(loader)		1025d	1412c
LIBRARIAN		1685c	2617c
gmaxheight			
C_HOOK	***	68d	
DGL_RAS(dgl_raster)	***	17369d	
GCRT(crtb)	***	141d	
gmem			
CRT(crt)		227d	255c

[illegible]

[illegible]

[illegible]

hp8280x.mp									
CTABLE(ctr)		528d	544c						
hp913x.a									
AMIGO(amigodvr)		689c	958c						
AMIGO(csamigo)		44d	165c						
CTABLE		1256c	1399c	1533c	1658s	1663s			
CTABLE(brstuff)	***	991d							
CTABLE(ctr)		248d	547c	565c	727c				
MINIT(hminit)	***	491c							
hp913x.a.mp									
CTABLE(ctr)		531d	547c						
hp913x.a.pp									
CTABLE(ctr)	***	565c							
CTABLE(options)	***	172d							
hp913x.b									
AMIGO(csamigo)		44d	166c						
CTABLE		1256c	1399c	1533c	1668s	1673s			
CTABLE(brstuff)	***	992d							
CTABLE(ctr)		249d	548c	566c	727c				
hp913x.b.mp									
CTABLE(ctr)		532d	548c						
hp913x.b.pp									
CTABLE(ctr)	***	566c							
CTABLE(options)	***	174d							
hp913x.c									
AMIGO(amigodvr)		492d	689c	958c					
AMIGO(csamigo)		44d	167c						
CTABLE		1256c	1399c	1533c	1678s	1683s			
CTABLE(brstuff)	***	992d							
CTABLE(ctr)		250d	549c	567c	727c				
MINIT(hminit)	***	491c							
hp913x.c.mp									
CTABLE(ctr)		533d	549c						
hp913x.c.pp									
CTABLE(ctr)	***	567c							
CTABLE(options)	***	176d							
hp98253									
KERNEL(general_0)	***	1084c							
KERNEL(iodeclarations)	***	357d							
hp98259									
KERNEL(general_0)	***	1105c							
KERNEL(iodeclarations)	***	359d							
hp98620									
DMA_DRV(init_dma)	***	236c							
KERNEL(iodeclarations)	***	343d							
hp98622									
F9885(f9885dvr)	***	262c							
G_DRV(init_gpio)		237c	293c						
KERNEL(general_0)	***	1053c							
KERNEL(iodeclarations)	***	352d							
hp98623									
KERNEL(iodeclarations)	***	353d							
hp98624									
H_DRV(init_hpb)		258c	304c						
KERNEL(general_0)	***	1039c							
KERNEL(iodeclarations)	***	350d							
hp98625									
DISCHPB(dischpb)		233c	240c	280c	339c	343c	483c	510c	
DI_DRV(init_discnt)		242c	281c						
KERNEL(general_0)	***	1060c							
KERNEL(iodeclarations)	***	354d							

hp98625									
IOLIB(serial_0)		2260c	2323c	2417c					
IOLIB(serial_3)		2571c	2636c	2711c	2765c	2817c	2852c		
KERNEL(general_0)	***	1045c							
KERNEL(iodeclarations)	***	351d							
PRINTER(prtdvr)	***	224c							
RS_DRV(init_rs)		247c	278c						
hp98627									
KERNEL(general_0)	***	1090c							
KERNEL(iodeclarations)	***	358d							
hp98627a.address									
DGL_C_OUT(dgl_conf_out)		11024d	11323c	11324c					
hp98628.async									
DC_DRV(intdc)		631c	679c						
DC_DRV(intdc)		276c	391c						
IOLIB(serial_0)		2240c	2303c	2369c					
IOLIB(serial_3)		2560c	2607c	2703c	2745c	2810c	2845c		
KERNEL(iodeclarations)	***	355d							
hp98628.dsnd1									
DC_DRV(intdc)		632c	680c						
DC_DRV(intdc)	***	277c							
KERNEL(iodeclarations)	***	340d							
hp98629									
DC_DRV(intdc)		633c	686c						
DC_DRV(intdc)	***	293c							
KERNEL(iodeclarations)	***	341d							
SRMMH(srmamodule)	***	150c							
hp98644									
IOLIB(serial_0)		2260c	2323c	2417c					
IOLIB(serial_3)		2572c	2637c	2712c	2766c	2817c	2852c		
KERNEL(general_0)	***	1140c							
KERNEL(iodeclarations)	***	360d							
RS_DRV(intdc)		247c	278c						
hp9885									
CTABLE		1195d	1254c	1396c					
CTABLE(brstuff)	***	991d							
CTABLE(ctr)		246d	545c	693c	694c	787c			
hp9885.default_dav									
CTABLE	***	1195d							
CTABLE(options)	***	117d							
hp9885.default_msus									
CTABLE		1194d	1458c						
hp9885.mp									
CTABLE(ctr)		529d	545c						
hp9895									
AMIGO(amigodvr)		482d	659c						
AMIGO(csamigo)		44d	163c	208c	316c				
CTABLE		1256c	1347c	1402c	1528c				
CTABLE(brstuff)	***	991d							
CTABLE(ctr)	***	247d	546c	706c	707c	798c			
MINIT(hminit)	***	490c							
hp9895.mp									
CTABLE(ctr)		530d	546c						
hp_datacomm									
DC_DRV(intdc)		635c	639c	671c					
DC_DRV(intdc)		275c	392c						
KERNEL(general_0)	***	1130c							
KERNEL(iodeclarations)	***	342d							
hp9ato									
KERNEL(general_0)	***	1078c							
KERNEL(iodeclarations)	***	356d							
hp9l_buffer_mode									
GLE_HPGL(gle_hp9l_out)		7384d	7546c						

[illegible]

11-82


```

ipadrequest
  AMIG0(amigodvr)      *** 639c
  ASCII(asciimodule)   *** 309c
  BUBBLES(bubble)      *** 118c
  CRT(crt)             *** 498c
  CS80(cs80dvr)        *** 1399c
  ETU                  *** 629c
  F988S(f9885dvr)      *** 290c
  GCRT(crtb)           *** 366c
  INIT(misc)           *** 343c
  INITLOAD(bootdammodule) *** 726c
  INITLOAD(mini)        *** 518c
  INITLOAD(sysglobals) *** 193d
  KEYS(keys)           *** 138c
  LIFDAM(lifmodule)     *** 488c
  M68KSYS(ci)          *** 741c
  SRMDAM(srmamodule)    *** 212c
  SRMDAM(srmamodule)    *** 1798c
  UCSD_AM(ucsd_am)      *** 263c
  UCSD_DAM(ucsdmodule)  *** 476c
ibadtile
  ETU                  *** 329c
  INIT(misc)           *** 311c
  INITLOAD(bootdammodule) *** 621c
  INITLOAD(sysglobals) *** 185d
  LIFDAM(lifmodule)     *** 103c
  SRMDAM(srmamodule)    *** 124c
  SRM_DRV(srm)          *** 1142c
  UCSD_DAM(ucsdmodule)  *** 175c
ibadunit
  GLE_KNOB(gle_knob_in) *** 18035c
  INIT(misc)           *** 306c
  INITLOAD(sysglobals) *** 184d
  SRM_DRV(srm)          *** 1136c
  UNIT0(uio)           *** 42c
ibadvalue
  ETU                  *** 163c
  INIT(misc)           *** 341c
  INITLOAD(sysglobals) *** 192d
  SRMDAM(srmamodule)    *** 233c
  SRM_DRV(srm)          *** 1144c
  UCSD_DAM(ucsdmodule)  *** 491c
icantsstretch
  INIT(misc)           *** 342c
  INITLOAD(sysglobals) *** 192d
icc
  CS80(cs80)           *** 401d
icode
  INIT(ldr)            *** 2409d
icuc
  CS80(cs80)           *** 416d
icuvai
  CS80(cs80)           *** 355d
  CS80(cs80dvr)        *** 1110c

```

```

id
  BUBBLES(bubble)      *** 140d
  CTABLE(ctr)          *** 336d
  EDRIIVER(edriver)    *** 63d
  GLE_RGL(gle_ras_out) *** 8378d
  HPHIL(hphil)         *** 257c
  INIT(cs801r)         *** 631d
  MIUI                 *** 395d
  SRM_DRV(srm)         *** 1931c
  SRM_DRV(srm)         *** 311d
  SYSDEVS(sysdevs)     *** 313d
idc_ident
  AMIG0(amigodvr)      *** 502d
idc_init
  DC_DRV(init_dc)      *** 614c
  DC_DRV(intdc)         *** 244d
idc_isr
  DC_DRV(init_dc)      *** 615c
  DC_DRV(intdc)         *** 245d
idc_rdt
  DC_DRV(init_dc)      *** 616c
  DC_DRV(intdc)         *** 246d
idc_rds
  DC_DRV(init_dc)      *** 620c
  DC_DRV(intdc)         *** 250d
idc_rdw
  DC_DRV(init_dc)      *** 618c
  DC_DRV(intdc)         *** 248d
idc_tr
  DC_DRV(init_dc)      *** 622c
  DC_DRV(intdc)         *** 254d
idc_wtb
  DC_DRV(init_dc)      *** 617c
  DC_DRV(intdc)         *** 247d
idc_wtc
  DC_DRV(init_dc)      *** 621c
  DC_DRV(intdc)         *** 252d
idc_wtw
  DC_DRV(init_dc)      *** 619c
  DC_DRV(intdc)         *** 249d
idchar
  EPROMS(eproms)       *** 26d
idcmd
  HPHIL(hphil)         *** 181d
ident
  AMIG0(amigodvr)      *** 428d
  CS80(cs80dvr)        *** 982d
  CTABLE(scanstuff)    *** 1052d
  DISCHPB(dischpb)     *** 200d
ident_table
  AMIG0(amigodvr)      *** 456d
ident_table_entries
  AMIG0(amigodvr)      *** 441d
ident_table_type
  AMIG0(amigodvr)      *** 452d
identity
  M68KSYS(ci)          *** 181d
idirfu
  INIT(misc)           *** 331c
  INITLOAD(sysglobals) *** 190d
  LIFDAM(lifmodule)     *** 882c
  UCSD_DAM(ucsdmodule) *** 216c

```


[illegible]

[illegible]


```

info4
DGL_HPGL(dgl_hppl)      *** 17200c
DGL_POLY(dgl_poly)      20081c 20088c 20995c
DGL_RAS(dgl_raster)     *** 17060c
GEN(dgl_gen)            *** 3544c
GLE_HPGL(gle_hppl_out)  7188c 7196c 7199c 7200c 7208c 7248c 7259c 7266c 7277c 7286c 7294c 7302c 7308c 7441c
GLE_HPGL(gle_hppl_in)   7682c
GLE_KNOB(gle_knob_in)   18114c 18122c 18125c 18126c 18154c 18164c 18173c 18181c 18189c 18197c 18203c 18388c
GLE_RGL(gle_ras_out)    18089c 18095c 18101c 18267c
GLE_SCLIP(gle_sclip)    *** 8187c 8510c
GLE_SMARK(gle_smark)    *** 6020c
GLE_TYPES(gle_types)    5119c 5170c
LIB(dgl_lib)            1024d 1200d
LIB(dgl_lib)            20984c 21005c 21133c 21345c

info_ptr1
DGL_C_OUT(dgl_config_out) 11236c 11258c 11324c
DGL_HPGL(dgl_hppl)       17045c 17049c 17177c
DGL_POLY(dgl_poly)       21051c 21096c
DGL_RAS(dgl_raster)      17218c 17281c 17332c
GLE_HPGL(gle_hppl_out)   7126c 7148c
GLE_HPGL(gle_hppl_in)    18071c 18086c
GLE_RGL(gle_ras_out)     8279c 8328c 8365c
GLE_SMARK(gle_smark)     *** 5139c
GLE_TYPES(gle_types)     *** 1026d 1202d
LIB(dgl_lib)             *** 20725c

info_ptr2
DGL_C_OUT(dgl_config_out) 11237c 11259c 11325c
DGL_POLY(dgl_poly)       21052c 21097c
GLE_TYPES(gle_types)     *** 1027d

info_ptr
BUBBLES(bubble)         57d 127c 145d
EDRIVER(edriver)        71d 103d

info_rec
M88KSYS5(ci)            44d 50d
info_recptr
M88KSYS5(ci)            50d 59d
inform_operator
PRINTER(prtdrvr)        165d 249c

info_start
LIBRARIAN
info_start
LIBRARIAN                83d 2737d 2747c 2825c 2953c 2975c 3008c 3124c
info_info
ETU                      78d 335c 388c 389c 391c 399c 885c 890c 918c 1023c 1228c
init
GLE_HPGL(gle_hppl_in)   *** 18240d
init_913x_aoc
MINIT(hminit)            409d 491c
init_9895_913x_chinook
MINIT(hminit)            332d 490c
init_aspect
DGL_VARS(dgl_vars)      *** 1182d
LIB(dgl_lib)             *** 21413c
init_bkgnd
DISCHPIX                 *** 26d
init_c
C_HOOK                   125d 145c
init_char_height
DGL_VARS(dgl_vars)      *** 1160d
LIB(dgl_lib)             *** 21435c
init_char_height_factor
DGL_VARS(dgl_vars)      *** 1162d
LIB(dgl_lib)             *** 21150c

init_char_rot_h
DGL_VARS(dgl_vars)      *** 1164d
LIB(dgl_lib)             21155c 21438c
init_char_rot_w
DGL_VARS(dgl_vars)      *** 1183d
LIB(dgl_lib)             21154c 21437c
init_char_width
DGL_VARS(dgl_vars)      *** 1159d
LIB(dgl_lib)             *** 21434c
init_char_width_factor
DGL_VARS(dgl_vars)      *** 1161d
LIB(dgl_lib)             *** 21448c
init_cmd_buf
MINIT(iramigo)           210d 215c
init_color
DGL_VARS(dgl_vars)      *** 1156d
LIB(dgl_lib)             21165c 21171c 21444c 21448c
init_color_table
DGL_RAS(dgl_raster)      17025d 17187c 17240c 17628c
init_color_table_def
DGL_RAS(dgl_raster)      17022d 17025d
init_cpx
DGL_VARS(dgl_vars)      *** 1167d
LIB(dgl_lib)             21176c 21426c 21429c
init_cpy
DGL_VARS(dgl_vars)      *** 1168d
LIB(dgl_lib)             21177c 21427c 21430c
init_dma
AMTGO(csamigo)           *** 56d
MINIT(iramigo)           *** 215c
init_dc
DC_DRV                    *** 718d
DC_DRV(init_dc)          *** 569d
init_dev_addr
DGL_VARS(dgl_vars)      *** 1170d
LIB(dgl_lib)             21409c 21411c
init_discint
DI_DRV                    *** 292d
DI_DRV(init_discint)     *** 178d
init_display_lm
DGL_VARS(dgl_vars)      *** 1187d
LIB(dgl_lib)             21040c 21419c
init_dma
DMA_DRV                   *** 285d
DMA_DRV(init_dma)        *** 152d
init_gpio
G_DRV                     *** 304d
G_DRV(init_gpio)         *** 192d
init_hpb
H_DRV                     *** 315d
H_DRV(init_hpb)          *** 193d
init_linestyle
DGL_VARS(dgl_vars)      *** 1157d
LIB(dgl_lib)             21163c 21172c 21445c
init_linewidth
DGL_VARS(dgl_vars)      *** 1158d
LIB(dgl_lib)             21173c 21446c
init_locator_lm
DGL_VARS(dgl_vars)      *** 1192d
LIB(dgl_lib)             21083c 21421c
init_root_password
SRM_DRV(srm)             *** 616d

```


init rs					
RS_DRV	***	291d			
RS_DRV(init rs)	***	189d			
init scanstuff					
CTAB.E	***	1438c			
CTAB.E(scanstuff)		1039d	1149d		
init segmenter					
SEGMENTER(asm)	***	75d			
SEGMENTER(segmenter)	***	24d			
init srm					
SRM.DM	***	26d			
init tapebuf					
CS80	***	1414c			
CS80(tapebuf)		49d	53d		
init timing_mode					
DGL_VARS(dgl_vars)	***	1166d			
LIB(dgl_lib)	***	21447c			
init u:sd_am					
UCSD_AH	***	281c			
UCSD_AH(ucsd_am)		32d	268d		
init u:sd_dam					
UCSD_DAP	***	687c			
UCSD_DAM(ucsdmodule)		34d	58d		
init viewport					
DGL_VARS(dgl_vars)	***	1177d			
LIB(dgl_lib)	***	21415c			
init vir_lim					
DGL_VARS(dgl_vars)	***	1184d			
LIB(dgl_lib)	***	21414c			
init window					
DGL_VARS(dgl_vars)	***	1172d			
LIB(dgl_lib)	***	21412c			
init with_d_bits					
MINI(hminit)		393c	447c		
MINI(iramigo)		191d	208d		
init a804x					
A804:DVR	***	416c			
A804:DVR(a804xdvr)		44d	377d	381c	407c
init ba					
BAT	***	30d			
init bootdam					
INIT initunits)	***	2162d			
INITI_OAD	***	1674c			
INITI_OAD(bootdammodule)		542d	730d		
init clock					
CLOCK	***	185c			
CLOCK(clock)		34d	173d		
init crt					
CRT	***	30d			
init crt'b					
GCR	***	29d			
init da'e					
MSKSYS(ci)		148d	649c	1182c	
init dig					
MINI(cs80ir)		634d	640c		
init dumbuf					
SRM(RV(srm))		1191d	1221c	1377c	1413c
init filekinds					
INIT	***	2550c			
INIT(misc)		214d	694d		
init files					
INIT		2521d	2570c		
init fklnds					
INITI_OAD		1702d	1781c		
init frames					
MSKSYS(ci)		579d	650c	1179c	
init head					
MSKSYS(ci)		941d	1167c		
init ial_pattern					
DGL_VARS(dgl_vars)	***	1218d			
init ial_seek					
AMIGO(amigodvr)		543d	627c	654d	
init ialize_bkgnd					
DISCHPIB	***	556c			
DISCHPIB(bkgnd)		61d	82d		
MIUI	***	544c			
init ialize_media					
MINI(cs80ir)		561d	649d	672c	
MINI(qminit)	***	1158c			
TAPEBKUP(cs80tbdv)	***	419c			
TAPEBKUP(cs80tbr)	***	25d	73d	96c	
init ialize_medium					
MIUI		283d	549c		
init ialize_options_byte					
MINI(qminit)		895d	1064c	1073c	1085c
TAPEBKUP		1246c	1477c	1478c	
TAPEBKUP(cs80tbdv)		328d	419c		
init ialize_data_transfer					
AMIGO(amigodvr)		546d	755c	756c	762d
init ialize_diagnostic					
MINI(cs80ir)		560d	625d	645c	
MINI(qminit)	***	1129c			
init ialize_transfer					
DISCHPIB(dischpio)		437d	500c	504d	
init ialize_rib					
MINI(xminit)		1375d	1520c	1830c	
init keys					
KEYS	***	598c			
KEYS(keys)		35d	577d		
init lang					
KEYS(keys)		231d	586c		
NONUS<BD1	***	212c			
NONUS<BD1(non_us_kbd1)		33d	74d	122c	204c
init link					
LIBRARY		2972d	3641c		
init loader					
INITI_OAD	***	2d			
init loader					
INITI_OAD	***	1782c			
INITI_OAD(loader)		1094d	1653d		
init loop					
HPHIL	***	389c			
HPHIL(hphil)		33d	353d	360c	373c
init media					
MINI(cs80ir)		661d	667c		
TAPEBKUP(cs80tbr)		85d	91c		
init mouse					
MOUSE	***	206c			
MOUSE(mouse)		33d	179d	196c	198c
init pag					
UCSD_DM(ucsd_am)		48d	97c	108c	109c
init sys					
INIT initunits)		217d	2258c		
init sysinit					
CTAB.E(ctr)	***	862c			
INIT	***	2565c			
INIT(dr)		2276d	2313d		

inittime															
CLOCK(clock)	126d	175c													
initunitable	2235d	2261c													
INIT(initunits)															
initunitentry															
INIT(initunits)	2181d	2240c	2243c	2245c	2247c	2249c									
initunits															
INIT	***	2514d													
INIT(initunits)	***	2133d													
M68KSYS(c1)		951d	1126c												
MINIT(xminit)	***	1359d													
initunit															
MINIT(cs80ir)		603d	613c	683d	693c	713d	719c	745d	751c	781d	792c	793c			
TAPEBKUP(cs80tbr)		267d	275c												
initvects															
INIT	***	2536c													
initworkfile															
M68KSYS(c1)		624d	651c	1173c											
inmount															
ETU		386d	899c												
inoaccess															
INIT(misc)	***	349c													
INITLOAD(sysglobals)	***	194d													
SRM_DRV(srm)	***	1162c													
inodirectory															
ETU	***	252c													
INIT(misc)	***	339c													
INITLOAD(sysglobals)	***	192d													
LIFDAM(lifmodule)		273c	494c												
MIUI	***	121d													
SRMDAM(srmdammodule)	***	835c													
TAPEBKUP	***	1052d													
UCSD_DAM(ucsdmodule)		117c	479c												
inoerror															
AMIGO(amigodvr)	571c	579c	595c	634c	648c	777c									
BUBBLE5(bubble)	40c	60c	110c												
CONVERT(convert_text)		51c	53c												
CRT(crt)	199c	257c	408c												
CS80(cs80dsr)	646c	653c	765c												
CS80(cs80dvr)	1119c	1135c	1232c	1333c	1342c	1356c	1368c	1382c	1387c						
CTABLE	***	1235c													
C_HOOD	***	110c													
DGL_RAS(dgl_raster)	17340c	17418c													
DISCHPIS(bkgnd)	***	118c													
EPROMS(eproms)	***	45c													
ETU		102c	116c	126c	133c	139c	203c	236c	268c	314c	358c	371c	467c	552c	599c
F9885(f9885dvr)		823c	905c	927c	1004c	1230c	1268c								
GCR(Crtb)		180c	185c	187c	293c										
GLE_KNOB(gle_knob_in)		186c	278c												
GLE_KNOB(gle_knob_in)	18051c	18065c													
INIT(misc)		304c	522c	549c	565c	573c									
INITLOAD(bootdammodule)		690c	754c												
INITLOAD(loader)	***	1121c													
INITLOAD(mini)		377c	400c	455c	521c										
INITLOAD(sysglobals)	***	184d													
KEYS(keys)	***	79c													
LIBRARYAN		134c	140c	289c	291c										
LIFDAM(lifmodule)		83c	253c	273c	493c	495c	1123c								
LOCKMOD(lockmodule)		39c	51c	54c	73c	80c	92c								
M68KSYS(c1)		161c	340c	422c	538c	600c	635c	641c	680c	753c	761c	829c	850c	1181c	
MINIT(bminit)		80c	87c												
MINIT(cs80ir)		763c	810c												
MINIT(hminit)		310c	501c												
MINIT(mminit)	***	168c													
MINIT(qminit)		952c	988c	997c	1001c	1140c	1277c								
MIUI		120d	381c												
MOREFSYS(mfs)		105c	385c	522c	527c	540c	545c	567c	674c						
PRINTER(prtdvr)	***	270c													
SRHAM(srmamodule)		73c	82c	88c	91c	120c	129c	135c	138c	156c	158c	183c			
SRMDAM(srmdammodule)		81c	245c	283c	479c	491c	522c	527c	531c	590c	733c	756c	759c	772c	790c
		798c	803c	809c	813c	828c	841c	854c	883c	923c	977c	1042c	1062c	1077c	1084c
		1089c	1095c	1111c	1117c	1120c	1128c	1131c	1141c	1144c	1153c	1186c	1188c	1199c	1225c
		1227c	1229c	1231c	1254c	1320c	1354c	1361c	1370c	1371c	1386c	1390c	1393c		


```

inotc_losed
  INIT(misc) *** 316c
  INITLOAD(sysglobals) *** 186d
  SRM_DRV(srm) *** 1172c
inotd_rect
  INIT(misc) *** 330c
  INITLOAD(sysglobals) *** 189d
inotlockable
  INIT(misc) *** 344c
  INITLOAD(sysglobals) *** 193d
  LOCKMOD(lockmodule) *** 42c 67c 83c
  SRMDAM(srmdammodule) *** 920c
inotondir
  INIT(misc) *** 352c
  INITLOAD(sysglobals) *** 195d
  SRMDAM(srmdammodule) *** 926c 1110c
  SRM_DRV(srm) *** 1165c
inotopen
  INIT(misc) *** 317c
  INITLOAD(sysglobals) *** 186d
  SRM_DRV(srm) *** 1154c
inotreadable
  INIT(misc) *** 328c
  INITLOAD(sysglobals) *** 189d
  MOREFSYS(mfs) *** 520c
inotvalidsize
  INIT(misc) *** 327c
  INITLOAD(sysglobals) *** 189d
  SRM_DRV(srm) *** 1138c
inotwriteable
  INIT(misc) *** 329c
  INITLOAD(sysglobals) *** 189d
inounit
  INIT(misc) *** 313c
  INITLOAD(bootdammodule) *** 719c
  INITLOAD(sysglobals) *** 185d
  MIU1 *** 144c
  SRMDAM(srmdammodule) *** 886c
  TAPEBKUP *** 1074c
inp
  SYSEVS(sysdevs) 240d 536c 543c 563c 640c
inpass
  SRMDAM(srmdammodule) 1529d 1537c 1553c 1557c 1564c 1572c
inpdtable
  MINIT(xminit) 1412d 1663c 1669c 1670c
input_cpx
  DGL_KNOB(dgl_knob) 18066c 18098c
  GLE_HPGLI(gle_hppl_in) 18258c 18390c
  GLE_KNOB(gle_knob_in) 18127c 18159c 18174c 18177c 18257c
  GLE_TYPES(gle_types) *** 1239d
input_cpy
  DGL_KNOB(dgl_knob) 18087c 18099c
  GLE_HPGLI(gle_hppl_in) 18259c 18391c
  GLE_KNOB(gle_knob_in) 18128c 18160c 18175c 18178c 18258c
  GLE_TYPES(gle_types) *** 1240d
input_echo
  GLE_GENI(gle_geni) *** 3052c
  GLE_HPGLI(gle_hppl_in) *** 18374c
  GLE_KNOB(gle_knob_in) *** 18224c
  GLE_TYPES(gle_types) *** 1206d
input_esc
  LIB(dgl_lib) 20056d 20750d

```

```

input_escapei
  GLE_GENI(gle_geni) *** 3058c
  GLE_HPGLI(gle_hppl_in) *** 18379c
  GLE_KNOB(gle_knob_in) *** 18225c
  GLE_TYPES(gle_types) *** 1207d
input_escapeg
  GLE_GENI(gle_geni) *** 3064c
  GLE_HPGLI(gle_hppl_in) *** 18376c
  GLE_KNOB(gle_knob_in) *** 18226c
  GLE_TYPES(gle_types) *** 1208d
input_handler_char_count
  GLE_GENI(gle_geni) *** 3026c
  GLE_HPGLI(gle_hppl_in) *** 18382c
  GLE_KNOB(gle_knob_in) *** 18219c
  GLE_TYPES(gle_types) *** 1230d
input_handler_name
  DGL_INQ(dgl_inq) *** 6546c
  GLE_HPGLI(gle_hppl_in) *** 18381c
  GLE_KNOB(gle_knob_in) *** 18218c
  GLE_TYPES(gle_types) *** 1229d
input_max
  DGL_KNOB(dgl_knob) *** 18082c 18095c
  GLE_HPGLI(gle_hppl_in) *** 18386c
  GLE_KNOB(gle_knob_in) 18087c 18174c 18265c
  GLE_TYPES(gle_types) *** 1235d
  LIB(dgl_lib) *** 21333c
input_max_y
  DGL_KNOB(dgl_knob) *** 18084c 18097c
  GLE_HPGLI(gle_hppl_in) *** 18388c
  GLE_KNOB(gle_knob_in) 18089c 18175c 18267c
  GLE_TYPES(gle_types) *** 1237d
  LIB(dgl_lib) *** 21335c
input_min_x
  DGL_KNOB(dgl_knob) 18081c 18094c
  GLE_HPGLI(gle_hppl_in) 18385c 18390c
  GLE_KNOB(gle_knob_in) 18086c 18174c 18264c
  GLE_TYPES(gle_types) *** 1234d
  LIB(dgl_lib) *** 21332c
input_min_y
  DGL_KNOB(dgl_knob) 18083c 18096c
  GLE_HPGLI(gle_hppl_in) 18387c 18391c
  GLE_KNOB(gle_knob_in) 18088c 18175c 18266c
  GLE_TYPES(gle_types) *** 1236d
  LIB(dgl_lib) *** 21334c
input_name
  DGL_INQ(dgl_inq) *** 6543c
  GLE_HPGLI(gle_hppl_in) 18140c 18141c 18142c 18158c 18159c 18167c 18168c 18176c 18184c 18192c 18356c 18357c 18363c
  GLE_KNOB(gle_knob_in) 18231c 18237c 18243c 18249c
  GLE_TYPES(gle_types) *** 1227d
input_name_char_count
  DGL_INQ(dgl_inq) *** 6540c 6542c 6544c
  GLE_GENI(gle_geni) *** 3025c
  GLE_HPGLI(gle_hppl_in) 18358c 18364c
  GLE_KNOB(gle_knob_in) 18232c 18238c 18244c 18250c
  GLE_TYPES(gle_types) *** 1228d
input_res_x
  DGL_INQ(dgl_inq) *** 6236c 6249c
  GLE_HPGLI(gle_hppl_in) *** 18378c
  GLE_KNOB(gle_knob_in) 18233c 18239c 18245c 18251c
  GLE_TYPES(gle_types) *** 1232d
  LIB(dgl_lib) 20281c 20907c 20908c

```


[illegible]

instr:uf		426d	461c	465c	468c	496c	500c	501c	503c	504c	505c	506c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c	540c</
----------	--	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	--------

internal_mus															
CTABLE	***	1479c													
CTABLE(brstuff)		932d	997c												
interruptlevel															
GLE_KNOB(gle_knob_in)	***	18115d													
KEYS(keys)		74d	85c	83c	88c	109c	114c	132c							
interrupttable															
INIT(initunits)	***	2176c													
INIT(isr)		73c	75c	83c	90c	118c	119c								
INIT_LOAD(sysglobals)	***	252d													
M68KSYS(c1)	***	296c													
intfId															
MINIT(asmr)	***	1326d													
MINIT(xmininit)		1427c	1572c	1587c	1630c	1681c	1702c	1724c	1734c	1759c	1800c				
intlevel															
A804XDVR(a804xdvr)	***	117c													
HPhIL(hphil)		138c	165c	187c	214c										
INIT(isr)	***	142c													
KEYS(keys)		85c	109c												
PRINTER(prtdvr)		177c	194c												
intlve															
MINIT(hminit)		254d	299d	399c	424c										
MINIT(iramigo)		192d	221d	236c											
intr_if															
KERNEL(iodeclarations)	***	560d													
intr_block_offset															
C580(csB0Dvr)		1220d	1245c	1249c	1257c	1265c	1268c	1291c	1295c						
integ															
BUBBLES(bubble)		136d	167c												
integaddr															
DC_DRV(init_dc)	***	648c													
DI_DRV(init_discint)	***	253c													
DMA_DRV(init_dma)		252c	256c												
G_DRV(init_gp10)	***	288c													
H_DRV(init_hp10)	***	276c													
INIT(isr)		69c	100c												
INIT_LOAD(sysglobals)	***	205d													
KERNEL(general_0)		1216c	1220c	1221c											
R5_DRV(init_rs)	***	256c													
integmask															
INIT(isr)		70c	101c												
INIT_LOAD(sysglobals)	***	206d													
integvalue															
INIT(isr)		71c	102c												
INIT_LOAD(sysglobals)	***	207d													
inttabletype															
INIT_LOAD(sysglobals)		213d	252d	275d											
invalid															
LIBRARIAN		417d	567c	621c	856c	861c	865c	869c	875c	889c	902c	905c	915c	921c	926c
		1005c	1029c	1076c											
invalidate_stateinfo															
C580(csB0Dsr)		602d	611d	701c											
MINIT(qminit)	***	1154c													
io_bit															
D1_DRV(extId1)		168d	169d	170d											
C_DRV(extId)		183d	184d	185d											
H_DRV(extId)		183d	184d	185d											
KERNEL(general_0)		839d	844d												
KERNEL(iodeclarations)		370d	434d	436d											

[illegible]


```

io temp_type3
  KERNEL(general_0)      *** 970d
  KERNEL(io_declarations) *** 520d
io term
  DGL_C_IN(dgl_conf_in) 12052c 12106c
  DGL_C_OUT(dgl_conf_out) 11278c 11434c 11475c
  GLE_GEN(gle_gen)      *** 2174c
  GLE_GENI(gle_geni)    *** 3046c
  GLE_TYPES(gle_types) 1067d 1217d
io tmp_ptr
  BUBBLES(bubble)       64c 67c 181c 162c
  DC_DRV(init_dc)       648c 652c 680c 678c 678c
  DISCHPIB(dischpib)    224c 250c 283c 278c 277c 278c 279c 282c 283c 298c 302c 303c 304c 319c
                        320c 321c 322c 335c 336c 337c 338c 341c 342c 352c 369c 370c 375c 376c
                        398c 399c 401c 402c 428c 466c 518c 531c 541c
  DI_DRV(init_discint)  247c 283c
  DMR_DRV(init_dma)     192c 209c
  EDRIVER(edriver)      *** 142c
  F9885(f9885dvr)       *** 264c
  GLE_HPIB(gle_hpib_io) 10068c 10077c 10100c 10113c 10122c 10124c 10202c 10206c 10208c 10232c 10290c 10294c 10296c 10320c
                        10520c 10543c
  G_DRV(init_gpio)      262c 295c
  H_DRV(init_hpib)       264c 306c
  IOLIB(general_1)       278c 287c 298c 310c 324c 349c
  IOLIB(general_2)       887c 869c 875c 877c 927c 950c
  IOLIB(general_4)       1338c 1363c 1391c 1402c 1405c 1416c 1419c 1451c 1476c 1522c 1550c 1670c
  IOLIB(hpib_0)          1727c 1738c 1751c
  IOLIB(hpib_1)          437c 448c 448c 502c 524c 528c 530c 554c 612c 616c 618c 642c 711c 761c
  IOLIB(hpib_3)          *** 2086c
  IOLIB(serial_0)        2281c 2344c 2466c
  KERNEL(general_0)      936c 952c 990c 1017c 1040c 1048c 1054c 1061c 1065c 1080c 1086c 1093c 1100c 1108c
                        1114c 1135c 1143c 1150c 1204c 1205c 1263c 1264c 1284c 1296c 1301c
  KERNEL(io_declarations) *** 540d
  PRINTER(prtdvr)       96c 111c 113c 115c 116c 137c 208c 217c 220c 221c 225c 231c 236c 288c
  RS_DRV(init_rs)        256c 260c 268c 281c
  SRH_DRV(srm)           1231c 1259c 1260c 1314c 1324c 1342c 1351c
io word
  DC_DRV(extdc)          205d 206d 207d 208d 209d 210d
  DC_DRV(init_dc)        *** 603d
  DC_DRV(intdc)          238d 248d 249d 250d 251d 252d 253d 264d 330d 340d 348d 349d 356d 357d
                        358d
  DI_DRV(extdi)          159d 160d 161d 162d 163d 164d
  DMR_DRV(init_dma)      183d 200d
  G_DRV(extg)            178d 177d 178d 179d 180d 181d
  H_DRV(exth)            174d 175d 176c 177d 178d 179d
  IOLIB(general_1)       306d 319d
  KERNEL(general_0)      748d 749d 751d 752d 754d 757d 760d 761d 763d 764d 824d 829d 834d 853d
                        854d 855d 872d 873d 874d 891d 910d 930d 931d 932d 945d 946d 947d
  KERNEL(io_declarations) *** 372d 381d 382d 424d 426d 428d 429d 431d 432d 502d 516d 530d
  RS_DRV(rs)             220d
                        176d 177d 178d 179d 180d 181d
io work_char
  IOLIB(general_2)       883c 970d 1018c 1052d 1061c 1062c 1074d 1082c
  IOLIB(hpib_3)          2097d 2101c 2104c
  KERNEL(io_declarations) *** 605d
io work_str
  IOLIB(general_2)       834d 862c 867c 868c 869c 870c 875c 876c 877c 878c 882c 883c 887c 918d
                        925c 927c
io work_word
  DMR_DRV(init_dma)      183d 188c 200d 205c

```

```

io write
  DGL_C_IN(dgl_conf_in)  *** 12104c
  DGL_C_OUT(dgl_conf_out) 11277c 11433c 11473c
  GLE_HPGL(gle_hppl_out) *** 7047c
  GLE_HPGLI(gle_hppl_in) *** 18034c
  GLE_TYPES(gle_types) 1068d 1218d
io buffer
  IOLIB(general_4)       1228d 1257d
io cb
  DGL_C_IN(dgl_conf_in) 12077c 12102c
  DGL_C_OUT(dgl_conf_out) 11384c 11390c 11432c 11472c
  GLE_GEN(gle_gen)       2130c 2174c
  GLE_GENI(gle_geni)     3028c 3046c
  GLE_HPGL(gle_hppl_out) 7047c 7147c 7180c 7240c 7566c 7567c 7573c 7581c 7610c
  GLE_HPGLI(gle_hppl_in) 18034c 18085c 18106c 18146c 18253c 18263c 18299c 18339c 18340c 18346c 18354c 18398c
  GLE_TYPES(gle_types) 1070d 1220d
io cb_ptr
  DGL_C_IN(dgl_conf_in) 12026d 12067d 12076c
  DGL_C_OUT(dgl_conf_out) 11047d 11167d 11367d 11382c 11389c
  GLE_FILE(gle_file_io) 9017d 9018d 9019d 9020d 9021d 9037d 9043d 9049d 9055c 9062d 9065c 9072d 9075c
  GLE_HPIB(gle_hpib_io) 10023d 10024d 10025d 10026d 10027d 10028d 10467d 10470c 10476d 10479c 10483d 10489c 10507d 10514c
  GLE_TYPES(gle_types) 10528d 10535c 10552d
                        1064d 1065d 1066d 1067d 1068d 1214d 1215d 1216d 1217d 1218d
io cb_ptr_hpib
  DGL_C_OUT(dgl_conf_out) 11370d 11384c 11402d 11431c 11432c 11437c 11443c 11453c
io cb_ptr
  DGL_C_IN(dgl_conf_in) 12070d 12077c 12087d 12101c 12102c 12109c 12113c 12119c
  DGL_C_OUT(dgl_conf_out) 11371d 11390c 11403d 11471c 11472c 11478c 11483c 11484c 11489c
io check
  CONVERT(convert_text) 49d 67c 72c 76c 79c
io code
  ETU                     119d 120c
io comasm
  F9885(f9885dvr)       *** 37d
  GLE_HPIB(gle_hpib_io) *** 10008d
  IOLIB(hpib_1)          *** 419d
  IOLIB(hpib_3)          *** 2053d
  IOLIB(serial_0)        *** 2228d
  IOLIB(serial_3)        *** 2524d
  KERNEL(general_0)       *** 772d
  KERNEL(iocomasm)       *** 687d
io control
  IOLIB(hpib_3)          *** 2123c
  IOLIB(serial_0)        2276c 2318c 2339c
  IOLIB(serial_3)        2565c 2566c 2577c 2632c 2665c 2706c 2717c 2760c 2783c 2813c 2820c 2848c 2855c
  KERNEL(general_0)       762d 944d
io clr
  DI_DRV(init_discint)   *** 232c
  G_DRV(init_gpio)       *** 247c
  H_DRV(init_hpib)       *** 248c
  IOLIB(hpib_0)          *** 1737c
  IOLIB(serial_0)        *** 2343c
  KERNEL(general_0)       *** 1192c
  KERNEL(io_declarations) *** 457d
io end
  DISCHPIB(dischpib)    370c 399c 531c
  DI_DRV(init_discint)   *** 227c
  H_DRV(init_hpib)       *** 243c
  IOLIB(hpib_1)          *** 501c
  KERNEL(general_0)       *** 1187c
  KERNEL(io_declarations) *** 452d

```



```

iod_init
DC_DRV(init_dc) *** 614c
DISCHP(B(dischp)) *** 224c
DI_DRV(init_discrnt) *** 219c 283c
GLE_HP(B(gle_hp)_io) *** 10067c
G_DRV(init_gpio) *** 237c 295c
H_DRV(init_hpib) *** 235c 306c
IOLIB(general_1) *** 277c
IOLIB(general_4) *** 1337c
KERNEL(general_0) *** 1175c 1235c 1281c 1283c
KERNEL(ioclarations) *** 444d
PRINTER(prtdvr) *** 96c 137c
RS_DRV(init_rs) *** 230c 281c
SRV_DRV(srm) *** 1231c

iod_isr
DC_DRV(init_dc) *** 615c 655c
DI_DRV(init_discrnt) *** 220c 260c
G_DRV(init_gpio) *** 238c 275c
H_DRV(init_hpib) *** 236c 283c
IOLIB(general_0) *** 1180c
KERNEL(ioclarations) *** 445d
RS_DRV(init_rs) *** 231c 263c

iod_poll
DISCHP(B(dischp)) *** 263c 283c 342c 428c
DI_DRV(init_discrnt) *** 230c
H_DRV(init_hpib) *** 246c
IOLIB(hpib_3) *** 2085c
KERNEL(general_0) *** 1190c
KERNEL(ioclarations) *** 455d

iod_rdb
DC_DRV(init_dc) *** 616c
DISCHP(B(dischp)) *** 369c 375c 398c 401c
DI_DRV(init_discrnt) *** 221c
GLE_HP(B(gle_hp)_io) *** 10543c
G_DRV(init_gpio) *** 239c
H_DRV(init_hpib) *** 237c
IOLIB(general_1) *** 286c
IOLIB(general_2) *** 867c 869c 875c 877c 950c 981c 1061c 1082c
KERNEL(general_0) *** 1181c 1233c
KERNEL(ioclarations) *** 446d
RS_DRV(init_rs) *** 232c

iod_rds
DC_DRV(init_dc) *** 620c
DI_DRV(init_discrnt) *** 225c
G_DRV(init_gpio) *** 243c
H_DRV(init_hpib) *** 241c
IOLIB(hpib_0) *** 935c 1185c
KERNEL(general_0) *** 450d
RS_DRV(init_rs) *** 236c

iod_rdw
DC_DRV(init_dc) *** 618c
DI_DRV(init_discrnt) *** 223c
G_DRV(init_gpio) *** 241c
H_DRV(init_hpib) *** 239c
IOLIB(general_1) *** 309c
KERNEL(general_0) *** 1183c
KERNEL(ioclarations) *** 448d
RS_DRV(init_rs) *** 234c

```

```

iod_send
DISCHP(B(dischp)) 276c 278c 279c 282c 304c 321c 322c 335c 337c 338c 341c 352c 376c 402c
DI_DRV(init_discrnt) *** 229c
GLE_HP(B(gle_hp)_io) *** 10112c
H_DRV(init_hpib) *** 245c
IOLIB(hpib_1) *** 436c
KERNEL(general_0) *** 1189c
KERNEL(ioclarations) *** 454d
PRINTER(prtdvr) *** 113c 115c 116c 217c 220c 221c 236c

iod_set
DISCHP(B(dischp)) *** 302c 319c
DI_DRV(init_discrnt) *** 231c
G_DRV(init_gpio) *** 246c
H_DRV(init_hpib) *** 247c
IOLIB(hpib_0) *** 1726c
IOLIB(serial_0) *** 2280c
KERNEL(general_0) *** 1191c
KERNEL(ioclarations) *** 456d

iod_temp
KERNEL(general_0) 780d 795d
KERNEL(ioclarations) 657d 672d 675c 680c

iod_test
DI_DRV(init_discrnt) *** 233c
G_DRV(init_gpio) *** 248c
H_DRV(init_hpib) *** 249c
IOLIB(hpib_0) *** 1750c
IOLIB(serial_0) *** 2465c
KERNEL(general_0) *** 1193c
KERNEL(ioclarations) *** 459d

iod_tfr
DC_DRV(init_dc) *** 622c
DISCHP(B(dischp)) *** 518c
DI_DRV(init_discrnt) *** 228c
G_DRV(init_gpio) *** 245c
H_DRV(init_hpib) *** 244c
IOLIB(general_4) *** 1475c 1521c 1549c
KERNEL(general_0) *** 1183c
KERNEL(ioclarations) *** 453d
RS_DRV(init_rs) *** 238c
SRV_DRV(srm) *** 1351c

iod_wto
DC_DRV(init_dc) *** 617c
DISCHP(B(dischp)) *** 298c 303c 320c
DI_DRV(init_discrnt) *** 222c
GLE_HP(B(gle_hp)_io) *** 10076c 10520c
G_DRV(init_gpio) *** 240c
H_DRV(init_hpib) *** 238c
IOLIB(general_1) *** 297c
IOLIB(general_2) *** 927c 1003c
KERNEL(general_0) *** 1182c 1234c
KERNEL(ioclarations) *** 447d
PRINTER(prtdvr) *** 231c
RS_DRV(init_rs) *** 233c

iod_wt:
DC_DRV(init_dc) *** 621c
DI_DRV(init_discrnt) *** 226c
G_DRV(init_gpio) *** 244c
H_DRV(init_hpib) *** 242c
KERNEL(general_0) *** 951c 1186c
KERNEL(ioclarations) *** 451d
PRINTER(prtdvr) *** 225c
RS_DRV(init_rs) *** 237c

```


ioe_wtw			
DC_DRV(init_dc)	***	619c	
DI_DRV(init_discint)	***	224c	
G_DRV(init_gpio)	***	242c	
H_DRV(init_hpib)	***	240c	
IOLIB(general_1)	***	323c	
KERNEL(general_0)	***	1184c	
KERNEL(iodeclarations)	***	449d	
RS_DRV(init_rs)	***	235c	
iodeclarations			
BUBBLES(bubble)	***	26d	
DC_DRV(extdc)	***	194d	
DC_DRV(init_dc)	***	579d	
DC_DRV(intdc)	***	233d	
DGL_C_OUT(dgl_conf_out)	***	11018d	
DISCHP18(bkgnd)	***	31d	
DISCHP18(dischp18)	***	189d	
DI_DRV(extdi)	***	151d	
DI_DRV(init_discint)	***	187d	
DMA_DRV(init_dma)	***	164d	
EDRIVER(edriver)	***	24d	
F9885(f9885dvr)	***	37d	
GLE_HP18(gle_hp18_io)	***	10033d	
G_DRV(extg)	***	168d	
G_DRV(init_gpio)	***	203d	
H_DRV(exth)	***	166d	
H_DRV(init_hpib)	***	203d	
IOLIB(general_1)	***	231d	
IOLIB(general_2)	***	793d	
IOLIB(general_3)	***	1103d	
IOLIB(general_4)	***	1192d	
IOLIB(hpib_0)	***	1703d	
IOLIB(hpib_1)	***	388d	
IOLIB(hpib_2)	***	1773d	
IOLIB(hpib_3)	***	2023d	
IOLIB(serial_0)	***	2208d	
IOLIB(serial_3)	***	2497d	
KERNEL(general_0)	***	736d	
KERNEL(iocomasm)	***	697d	
KERNEL(iodeclarations)	***	222d	
LI8(dgl_lib)	***	20111d	
MUI	***	11d	
PRINTER(prtdvr)	***	36d	
RS_DRV(init_rs)	***	197d	
RS_DRV(rs)	***	167d	
SRAMH(srammodule)	***	34d	
SRMDH(srmdanmodule)	***	36d	
SRM_DRV(srm)	***	29d	
ioe_bad_cnt			
IOLIB(general_3)	***	1131c	
IOLIB(general_4)	***	1374c	
KERNEL(iodeclarations)	***	265d	
SRM_DRV(srm)	***	1317c	
ioe_bad_sct			
IOLIB(general_3)	***	1141c	
IOLIB(serial_0)	***	2238c	2250c 2270c 2301c 2313c 2333c 2367c 2408c 2456c
KERNEL(iodeclarations)	***	275d	
ioe_bad_tfr			
DC_DRV(intdc)	***	429c	
IOLIB(general_3)	***	1128c	
IOLIB(general_4)	***	1499c	1514c
KERNEL(iodeclarations)	***	262d	

ioe_bad_tmo			
GLE_HP18(gle_hp18_io)	10087c	10092c	
IOLIB(general_1)	***	336c	341c
IOLIB(general_3)	***	1132c	
KERNEL(iodeclarations)	***	266d	
ioe_buf_busy			
IOLIB(general_3)	***	1130c	
IOLIB(general_4)	***	1307c	1381c 1573c 1600c
KERNEL(iodeclarations)	***	264d	
SRM_DRV(srm)	***	1276c	1319c
ioe_crd_down			
IOLIB(general_3)	***	1142c	
KERNEL(iodeclarations)	***	276d	
ioe_dc_act			
IOLIB(general_3)	***	1161c	
KERNEL(iodeclarations)	***	288d	
ioe_dc_car			
IOLIB(general_3)	***	1160c	
KERNEL(iodeclarations)	***	287d	
ioe_dc_clk			
IOLIB(general_3)	***	1158c	
KERNEL(iodeclarations)	***	285d	
ioe_dc_conf			
IOLIB(general_3)	***	1163c	
KERNEL(iodeclarations)	***	290d	
ioe_dc_conn			
IOLIB(general_3)	***	1162c	
KERNEL(iodeclarations)	***	289d	
ioe_dc_cts			
IOLIB(general_3)	***	1159c	
KERNEL(iodeclarations)	***	286d	
ioe_dc_fail			
IOLIB(general_3)	***	1148c	1155c
KERNEL(iodeclarations)	***	282d	
ioe_dc_ovfl			
IOLIB(general_3)	***	1157c	
KERNEL(iodeclarations)	***	284d	
ioe_dc_reg			
IOLIB(general_3)	***	1164c	
KERNEL(iodeclarations)	***	291d	
ioe_dc_rval			
IOLIB(general_3)	***	1149c	1165c
KERNEL(iodeclarations)	***	292d	
ioe_dc_uart			
IOLIB(general_3)	***	1156c	
KERNEL(iodeclarations)	***	283d	
ioe_eod_seen			
DC_DRV(intdc)	***	472c	504c
IOLIB(general_3)	***	1143c	
KERNEL(iodeclarations)	***	277d	
ioe_isc			
DC_DRV(init_dc)	***	697c	
DC_DRV(intdc)	***	473c	505c
DISCHP18(dischp18)	***	221c	
KERNEL(iodeclarations)	***	611s	666c
PRINTER(prtdvr)	***	92c	98c 100c 102c 139c
ioe_isc_busy			
IOLIB(general_3)	***	1129c	
IOLIB(general_4)	***	1394c	1408c
KERNEL(iodeclarations)	***	263d	
SRM_DRV(srm)	***	1328c	1336c

II-98

10minisc														
BUBBLES(bubble)	***	149c												
DC_DRV(init_dc)		628c	669c											
DI_DRV(init_discint)		239c	279c											
EDRIVER(edriver)	***	131c												
G_DRV(init_gprio)		254c	291c											
H_DRV(init_hpib)		255c	302c											
KERNEL(general_0)		984c	1199c	1258c	1278c									
KERNEL(iodeclarations)		241d	388d	369d										
RS_DRV(init_rs)		244c	276c											
10r														
A804XQVR(a804xdvr)	***	165c												
DISCHPIB(bkgnd)		65d	155d	156c										
EDRIVER(edriver)	***	285c												
INITLOAD(loader)		1591d	1646c	1649c										
INITLOAD(mini)		414d	454c	455c	464c	468c	472c							
10read_byte														
DC_DRV(init_cc)		647c	683c											
DC_DRV(intdc)	***	230c												
DI_DRV(init_discint)	***	248c												
G_DRV(init_gprio)	***	267c												
H_DRV(init_hpib)	***	270c												
KERNEL(general_0)		753d	890d	903c	1032c	1124c	1127c							
RS_DRV(init_rs)	***	255c												
10read_word														
DMA_DRV(init_dma)		188c	205c	234c										
KERNEL(general_0)		747d	852d	865c										
10res														
AMIGO(amigodvr)	***	777c												
CS80(cs80dvr)	***	1119c												
DISCHPIB(bkgnd)		39d	118c	133c	158c									
MINIT(cs80ir)		761c	763c	808c	810c									
MINIT(hminit)		309c	310c											
MINIT(qminit)		986c	988c	1134c	1140c									
MUI		23d	26c											
TAPEBKUP		987d	990c											
TAPEBKUP(cs80tbdvr)		736c	739c	746c	748c									
TAPEBKUP(cs80tbr)		286c	289c											
10res_value														
F9885(f9885dvr)		94d	100c											
10resc														
CS80(cs80dvr)		1238c	1242c	1248c										
F9885(f9885dvr)		89c	119c	138c	156c	180c	217c	220c	224c	228c	235c	236c	237c	242c
INITLOAD(mini)		280c	283c	285c	290c	293c								
MINIT(mminit)	***	311d	322d	401c	468c	472c	490c	493c	508c	511c	513c	518c	521c	
PRINTER(prtdvr)		168c												
10resc_bkgnd		105c	132c	140c	199c	290c	312c							
10resc_bkgnd														
AMIGO(amigodvr)		470c	477c	478c	483c	516c	519c	528c	529c	533c	535c	605c	607c	611c
CS80(cs80dvr)		668c	671c	802c	828c	873c	874c	880c	882c	889c	891c	892c	893c	899c
CS80(cs80dvr)		903c	906c	910c	914c	922c	923c	926c	931c	943c				
DISCHPIB(bkgnd)		172c	212c											
DISCHPIB(bkgnd)		651c	756c	766c	923c									
DISCHPIB(bkgnd)		1005c	1104c	1129c										
DISCHPIB(dischpib)		65d	155d											
MINIT(hminit)		222c	225c	249c	353c									
MINIT(qminit)		325c	327c	361c	379c	395c	400c	405c	426c	449c	460c	474c	475c	481c
TAPEBKUP(cs80tbdvr)		482c	492c											
TAPEBKUP(cs80tbr)	***	958c	1137c	1254c										
10reset		351c	531c	646c	656c	684c	759c	879c	889c					
GLE_HPIB(gle_hpib_io)		29d												
10LIB(general_1)		10064d	10498c											
		237d	274d											
10result_value														
INITLOAD(mini)		311d	322d	324c										
10rs1twd														
CS80(cs80dsr)		638d	639d											
DISCHPIB(bkgnd)		39d	65d	155d										
ETU	***	119d												
INITLOAD(bootdamodule)	***	594d												
INITLOAD(mini)		311d	313d	322d	329d	335d	414d							
INITLOAD(sysglobals)	***	178d												
LIFDAM(lifmodule)	***	86d												
TAPEBKUP(cs80tbdvr)		518d	519d	698d	699d									
10rval														
INITLOAD(mini)		313d	329d	377c	380c	400c	401c	454c						
MINIT(mminit)	***	168c												
10rval to report														
CS80(cs80dsr)		638d	646c	751c	752c	765c	766c							
TAPEBKUP(cs80tbdvr)		518d	526c	632c	633c	655c	656c	698d	738c	739c	754c	863c	864c	888c
10s_access_to_file_not_allowed														
SRM_DRV(srm)		959d	1162c											
10s_attach_table_full														
SRM_DRV(srm)		943d	1137c											
10s_bad_file_name														
SRM_DRV(srm)		962d	1141c											
10s_bad_select_code														
SRM_DRV(srm)		937d	1136c											
10s_conflicting_share_modes														
SRM_DRV(srm)		961d	1168c											
10s_corrupt_directory														
SRM_DRV(srm)	***	952d												
10s_deadlock_detected														
SRM_DRV(srm)		977d	1167c											
10s_device_drivers_dont_match														
SRM_DRV(srm)	***	941d												
10s_directory_formats_dont_match														
SRM_DRV(srm)	***	945d												
10s_directory_not_empty														
SRM_DRV(srm)		971d	1183c											
10s_d5_rom_messaging														
SRM_DRV(srm)	***	939d												
10s_duplicate_filenames														
SRM_DRV(srm)		965d	1175c											
10s_duplicate_passwords														
SRM_DRV(srm)		976d	1160c											
10s_eof_encountered														
SRM_DRV(srm)		981d	1178c											
10s_error_base														
SRM_DRV(srm)	***	935d												
10s_error_top														
SRM_DRV(srm)	***	987d												
10s_file_in_use														
SRM_DRV(srm)		963d	1171c											
10s_file_locked_please_retry														
SRM_DRV(srm)		983d	1169c											
10s_file_not_directory														
SRM_DRV(srm)		970d	1182c											
10s_file_not_found														
SRM_DRV(srm)		968d	1180c											
10s_file_pathname_missing														
SRM_DRV(srm)		950d	1142c											
10s_file_unopened														
SRM_DRV(srm)		955d	1154c											
10s_illegal_byte_number														
SRM_DRV(srm)		951d	1144c											

ios_improper_mass_storage_device		
SRM_DRV(srm)	***	944d
ios_insufficient_disk_space		
SRM_DRV(srm)		964d 1174c
ios_invalid_file_code		
SRM_DRV(srm)		982d 1184c
ios_invalid_file_id		
SRM_DRV(srm)		947d 1139c
ios_invalid_file_size		
SRM_DRV(srm)		946d 1138c
ios_invalid_ios_request		
SRM_DRV(srm)	***	942d
ios_invalid_protect_code		
SRM_DRV(srm)		973d 1158c
ios_link_to_directory_not_allowed		
SRM_DRV(srm)		978d 1165c
ios_no_capability_for_file		
SRM_DRV(srm)		967d 1157c
ios_no_reply		
SRM_DRV(srm)		984d 1147c
ios_password_not_allowed		
SRM_DRV(srm)		958d 1156c
ios_password_not_found		
SRM_DRV(srm)		975d 1159c
ios_phys_eof_encountered		
SRM_DRV(srm)		966d 1177c
ios_purge_on_open		
SRM_DRV(srm)		985d 1172c
ios_rename_across_volumes		
SRM_DRV(srm)	***	979d
ios_software_bug		
SRM_DRV(srm)	***	936d
ios_successful_completion		
SRM_DRV(srm)		953d 1146c
ios_system_down		
SRM_DRV(srm)		954d 1149c
ios_unallocated_extent		
SRM_DRV(srm)	***	938d
ios_unsupported_dar		
SRM_DRV(srm)	***	940d
ios_unsupported_directory_operation		
SRM_DRV(srm)		960d 1164c
ios_volume_down		
SRM_DRV(srm)		980d 1152c
ios_volume_in_use		
SRM_DRV(srm)		969d 1181c
ios_volume_io_error		
SRM_DRV(srm)	***	949d
ios_volume_labels_dont_match		
SRM_DRV(srm)	***	957d
ios_volume_not_found		
SRM_DRV(srm)		972d 1151c
ios_volume_offline		
SRM_DRV(srm)		956d 1150c
ios_volume_recoverable_error		
SRM_DRV(srm)	***	948d
ios_volume_unrecoverable_error		
SRM_DRV(srm)	***	974d

iostatus														
GLE_HPIB(gle_hpib_io)	10148c	10243c	10252c	10259c	10261c	10331c	10340c	10347c	10349c					
IOLIB(hoib_1)	472c	487c	585c	574c	581c	583c	653c	662c	669c	671c				
IOLIB(hoib_3)	2138c	2148c	2157c	2167c										
IOLIB(serial_0)	2253c	2274c	2316c	2337c	2411c	2459c								
IOLIB(serial_3)	2646c	2663c	2715c	2781c										
KERNEL(general_0)	758d	828d	939c											
iotemp														
SEGMINTER(asm)	180d	213c	215c											
iounitialize														
IOLIB(general_1)	236d	265d												
iounit														
MINI(asmr)	1331d	1335d	1342d	1346d										
iowrite_byte														
KERNEL(general_0)	756d	909d												
iowrite_word														
KERNEL(general_0)	750d	871d												
ip														
M8KSYS(ci)	787d	808c												
SRM_DRV(srm)	1373d	1383c	1384c	1391c	1392c									
ipart														
CLOCK(clock)	40c	94c	95c	146c	150c	152c	156c							
ipointer														
INITLOAD(sysglobals)	***	31d												
M8KSYS(ci)	395d	738c	740c	787d										
iptr														
CRT(crt)	546d	633c	643c	645c										
GCRT(crtb)	416d	471c												
LIBRARIAN	1487d	1498c	1499c	1501c	1502c	1506c	1515c	1521c	1528c	1529c	1531c	1532c	1533c	
iramigo														
MINI(hminit)	***	248d												
MINI(iramigo)	***	184d												
irate														
SYSDEV(Sysdevs)	422d	424c												
ires														
LIBRARIAN	58d	93c	96c	3663c	3669c	3679c								
isc														
IOLIB(general_4)	1249d	1667d	1670c											
isc_busy														
IOLIB(general_4)	1249d	1667d	1672c	1673c										

isc table																
BUBBLES(bubble)	63c	64c	67c	150c												
DC_DRV(init_dc)	629c	670c														
DC_DRV(initdc)	274c	29c	390c													
DISCHPIS(dischpib)	223c	23c	233c	246c	261c	292c	315c	351c	363c	394c	426c	446c	508c	526c		
DI_DRV(init_discint)	240c	280c														
DMA_DRV(init_dma)	192c	209c	235c	264c												
EDRIVER(edriver)	132c	147c														
F9885(f9885dvr)	***	260c														
GLE_HPIB(gle_hpib_io)	10066c	10075c	10095c	10111c	10122c	10124c	10146c	10200c	10229c	10289c	10317c	10472c	10517c	10538c		
G_DRV(init_gp10)	259c	282c														
H_DRV(init_hpib)	256c	303c														
IOLIB(general_1)	276c	28c	295c	308c	322c	344c										
IOLIB(general_2)	857c	858c	922c	923c	943c	944c	974c	975c	999c	1000c	1057c	1058c	1078c	1079c		
IOLIB(general_4)	1336c	1363c	1391c	1402c	1405c	1416c	1419c	1449c	1451c	1474c	1476c	1520c	1522c	1548c		
IOLIB(hpib_0)	1550c	1670c														
IOLIB(hpib_1)	1725c	1736c	1743c													
IOLIB(hpib_3)	435c	446c	443c	470c	485c	500c	522c	551c	610c	639c	708c	758c				
IOLIB(serial_0)	2084c	2120c														
IOLIB(serial_3)	2236c	2240c	2299c	2303c	2365c	2369c										
KERNEL(general_0)	2556c	2560c	2571c	2572c	2603c	2607c	2636c	2637c	2687c	2703c	2711c	2712c	2741c	2745c		
KERNEL(iodeclarations)	2765c	2766c	2806c	2810c	2841c	2845c										
KERNEL(ioeclarations)	857c	876c	896c	915c	934c	950c	984c	1202c	1204c	1205c	1206c	1239c	1240c	1259c		
KERNEL(ioeclarations)	1279c	1296c	1301c													
MIUI	***	616d														
PRINTER(prtdvr)	***	350c														
RS_DRV(init_rs)	***	274c														
SRMAM(srmammodule)	***	245c	277c													
SRM_DRV(srm)	***	149c														
isc table type	1230c	1256c	1304c													
DISCHPIS(dischpib)	270d	289d	312d	329d	359d	385d	442d									
KERNEL(ioeclarations)	538d	617d														
PRINTER(prtdvr)	***	67d														
isc te_ptr	270d	274c	323d	333c												
DISCHPIS(dischpib)	289d	292c	293c	294c	312d	315c	316c	317c	359d	363c	364c	365c	385d	394c		
DISCHPIS(dischpib)	395c	396c	442d	446c	452c	457c	464c									
isize																
DGL_HPGL(dgl_hppl)	17017d	17062d														
DGL_INQ(dgl_inq)	6010d	6086d	6171c													
DGL_RAS(dgl_raster)	17348d	17427d														
DGL_VARS(dgl_vars)	1123d	1128d														
LIB(dgl_lib)	20057d	20064d	20751d	20763c	20764c	20788d	20781c	20782c								
isr																
A804XDVR(a804xdvr)	***	31d														
BUBBLES(bubble)	***	26d														
DC_DRV(init_dc)	***	595d														
DC_DRV(initdc)	***	259d														
DI_DRV(init_discint)	***	201d														
DMA_DRV(init_dma)	***	177d														
G_DRV(init_gp10)	***	219d														
H_DRV(init_hpib)	***	217d														
INIT(isr)	***	32d														
MINIT(xmininit)	***	1360d														
RS_DRV(init_rs)	***	213d														
ischange																
INIT(isr)	55d	136d														
isrib																
A804XDVR(a804xdvr)	***	64d														
BUBBLES(bubble)	***	32d	124d	129c												
INITLOAD(sysglobals)	***	201d	203d													
KERNEL(ioeclarations)	***	493d	507d	521d	599d	600d										
MINIT(xmininit)	***	1375d														
isribptr																
INIT(isr)	43d	50d	53d	56d	65d	67c	75c	83d	98c	107c	108c	112d	121c	137d		
INIT(isr)	144c															
isribptr																
A804XDVR(a804xdvr)	89d	115d	122c													
INIT(intunites)	2142d	2167d														
INITLOAD(sysglobals)	***	202d														
isrinfo																
BUBBLES(bubble)	144d	161c	170c													
isrlink																
INIT(isr)	38d	60d														
MINIT(xmininit)	***	1520c														
isrmcatchall																
INIT(misc)	***	354c														
INITLOAD(sysglobals)	***	195d														
SRMAM(srmammodule)	***	214c	220c													
SRMODM(srmammodule)	1812c	1819c														
SRM_DRV(srm)	1186c	1358c														
isrproctype																
INIT(isr)	38d	45d	55d	60d	78d	136d										
INITLOAD(sysglobals)	202d	209d														
KERNEL(general_0)	***	1180c														
KERNEL(ioeclarations)	***	445d														
isrrec																
A804XDVR(a804xdvr)	64d	122c	386c													
isrunlink																
DC_DRV(init_dc)	***	651c														
DI_DRV(init_discint)	***	256c														
DMA_DRV(init_dma)	***	253c	257c													
G_DRV(init_gp10)	***	271c														
H_DRV(init_hpib)	***	279c														
INIT(isr)	52d	111d														
MINIT(xmininit)	***	1830c														
RS_DRV(init_rs)	***	259c														
issue_cmd																
AMIGO(amigodvr)	***	753c														
AMIGO(csamigo)	***	144d														
MINIT(xmininit)	***	205c	215c	239c	310c	334c	388c	401c	412c							
issue transfer request																
AMIGO(amigodvr)	545d	705c	710d													
istrovfl																
INIT(misc)	***	332c														
INITLOAD(sysglobals)	***	190d														
IOLIB(general_2)	***	894c														
MOREFSYS(mfs)	***	542c														
italian_kbd																
A804XDVR(a804xdvr)	***	187d	164c													
NONUSKBD2	***	155c														
SYSEVS(sysevs)	***	148d														
1tet																
AMIGO(amigodvr)	447d	452d	457d	458d	459d	460d	461d	462d	463d							

iifkdb															
A804XVR(a804xdvr)		214c	224c	341c	399c										
HPHIL(hphil)		***	364c												
KEYS(keys)		***	310c												
MOUSE(mouse)		***	181c												
NONUSKBD1(non_us_kbd1)			54c	154c											
NONUSKBD2		***	127c												
SYSDEVS(sysdevs)		***	145d												
itcomanvopen															
INIT(misc)		***	348c												
INITLOAD(sysslba.s)		***	194d												
SRM_DRV(srm)		***	1137c												
iu															
CS80(cs80)		***	95d												
M8KS/S(c1)			856d	864c	865c										
ivalue															
SYSDEVS(sysdevs)			429d	430c											
ix															
DGL_POLY(dgl_poly)		20268d	20300c	20310c	20445d	20451c	20456c	20462c	20464c	20488c	20490c	20495c	20497c	20505c	20507c
LIB(dgl_lib)		20049d	20050d	20616d	20628c	20633c	20638c	20646d	20659c	20663c	20669c				
iy															
DGL_POLY(dgl_poly)		20268d	20301c	20311c	20445d	20451c	20456c	20463c	20465c	20489c	20491c	20496c	20498c	20506c	20508c
LIB(dgl_lib)		20049d	20050d	20616d	20629c	20634c	20638c	20646d	20660c	20664c	20669c				
j															
A804XVR(a804xdvr)		233d	236c												
ASCII(asciimodule)		101d	112c	114c	115c	116c									
CONVERT(convert_text)		46d	66c	68c	90d	110c	113c	114c	115c	117c	120c				
CRT(crt)		231d	254c	255c	551d	575c	579c	582c	584c	585c	588c	594c	596c	603c	605c
C_HOOK		607c	608c	623c	624c	714d	718c								
DGL_POLY(dgl_poly)		79d	95c	100c											
DGL_RNS(dgl_raster)		20411d	20775c	20776c	20779c	20780c	20784c	20786c	20787c	20790c	20792c	20793c	20795c	20801c	20804c
ETU		17380d	17403c	17408c											
GCRT(crtb)		852d	995c	998c	999c	1001c	1006c	1104c							
GEN(dgl_gen)		152d	171c	176c	372d	377c	409c	420d	470c	471c					
GLE_RSL(gle_ras_out)		3170d	3174c	3175c	3180c										
HEAPT(hom)		8153d	8205c	8206c											
INIT(f)		***	217d												
INIT(s)		***	747d												
INITLOAD(bootdammodule)		1737d	1747c	1751c	1753c										
LIBRAFIAN		614d	695c	696c	697c	699c	700c								
MINIT(xmininit)		109d	117c	118c	752d	787c	730c	796c	799c	1369d	3119d				
MIUI		1444d	1447c	1448c	1464d	1480c	1481c								
MORESYS(mfs)		377d	519c												
SRMDAM(srmdammodule)		564d	600c	610c	611c	612c	615c	618c	619c	620c	621c	622c	624c	625c	626c
UCSD_RH(ucsd_am)		467d	509d	575d											
UCSD_RH(ucsdmodule)		71d	78c	79c	80c	82c	83c	84c	87c						
UCSD_TAM(ucsdmodule)		136d	141c	148c	149c	152c	153c	531d	535c	536c	538c	569d	581c	594c	613c
617c		620c	621c												
jmplstat															
LIBRAFIAN		838d	920c	925c											
jptr															
LIBRAFIAN		1487d	1513c	1515c	1517c	1518c	1519c								
junkint															
LIBRAFIAN		317d	354c	355c	359c	360c	1210c	1210c							
k															
A804XVR(a804xdvr)		329c	330c	333d	334c										
CLOCK(clock)		50d	54c	55c											
CRT(crt)		551d	584c	585c	597c	600c	608c	609c	643c	644c	645c	664d			
LIBRAFIAN		471d	473c	474c	752d	789c	790c	796c	799c	1369d	3119d				
LIFDAM(lifmodule)		100c	104c	105c	361d	365c	366c	5c	379c						
MINIT(xmininit)		1453d	1467c												
MOUSE(mouse)		***	77d	114c	115c	116c	118c								
UCSD_TAM(ucsdmodule)		***	56d												
k1															
CLOCK(clock)			50d	55c	56c	57c									
k128															
EPROMS(eproms)		***	20d												
k16															
EPROMS(eproms)			22d	58c	66c										
ETU			35d	959c	978c	979c	981c	999c	1013c	1090c	1101c				
k2															
CLOCK(clock)			50d	58c	59c	60c									
NONUSKBD1(non_us_kbd1)			100d	101d	108d	109d									
k256															
EPROMS(eproms)			21d	107c											
kanaalphabet															
NONUSKBD1(non_us_kbd1)			105d	149c	150c										
kanatocrtlookup			62d	390c											
CRT(crt)			47d	62d											
kanatocrtlookuptype															
CRT(crt)															
kappend															
KEYS(keys)			368c	568c											
SYSDEVS(sysdevs)			232c	531c	534c										
kata															
GLE_TYPES(gle_types)		***	1118d												
LIB(dgl_lib)			21401c	21402c											
katakana_kbd															
A804XVR(a804xdvr)		***	192d	194d											
CRT(crt)		***	396c												
GCRT(crtb)		***	268c												
LIB(dgl_lib)		***	21401c												
NONUSKBD1(non_us_kbd1)			77d	115c	130c	143c	200c								
NONUSKBD2		***	200c												
SYSDEVS(sysdevs)		***	150d												
katakana_trans															
NONUSKBD1(non_us_kbd1)			37d	145c											
kbd															
C_HOOK		***	40d												
DGL_TOOLS(dgl_tools)		***	20018d												
kbd_crt_drivers															
KERNEL(general_0)			742d	1231c	1232c	1239c	1240c								
kbd_disable															
SYSDEVS(sysdevs)		***	139d												
kbd_enable															
SYSDEVS(sysdevs)		***	139d												
kbd_rdb															
KERNEL(general_0)			780d	1233c											
kbd_tm_name															
CTABLE(ctr)			359d	651c											
kbal110c_k															
A804XVR(a804xdvr)		***	401c												
KEYS(keys)		***	263c												
NONUSKBD1(non_us_kbd1)			49c	52c	56c	58c	200c								
NONUSKBD2		***	204c												
SYSDEVS(sysdevs)		***	289d	650c											
kdcars10c_k															
A804XVR(a804xdvr)		***	401c												
KEYS(keys)		***	216c	262c	480c										
NONUSKBD1(non_us_kbd1)		***	200c												
NONUSKBD2		***	204c												
SYSDEVS(sysdevs)		***	289d	650c											
kdcconf10c_k															
A804XVR(a804xdvr)		***	213c	214c	216c	218c									
SYSDEVS(sysdevs)		***	158d												
lbddata															
KEYS(keys)			355d	360c	362c	373d	427c								

II-102

kbdhooktype		37d													
R804XDVR(a804xdvr)		68d	38d												
HPHIL(hphil)	***	S5d													
KEYS(keys)	***	37d	155d	181d	224d	287d	333d	340d							
SYSDEVS(sysdevs)															
kbdio															
CRT(crt)		415c	423c	439c											
GCRTR(ctrb)		285c	293c	309c											
INIT(unitsunits)		2151d	2246c												
SYSDEVS(sysdevs)		372d	432d												
kbdiohook															
KEYS(keys)	***	582c													
SYSDEVS(sysdevs)		154d	434c	618c											
kbdishook															
R804XDVR(a804xdvr)	***	106c													
KEYS(keys)		352c	501c	502c	580c										
MOUSE(mouse)		111c	119c												
SYSDEVS(sysdevs)		155d	615c												
kbdlang															
R804XDVR(a804xdvr)		223c	224c	225c											
CRT(crt)	***	396c													
GCRTR(ctrb)	***	266c													
LIB(dgl_l1b)	***	21401c													
NONUSKBD1(non_us_kbd1)		115c	117c	128c	130c	138c	139c	143c	175c	181c	200c				
NONUSKBD2		128c	137c	139c	140c	141c	142c	147c	148c	151c	164c	184c	205c	208c	211c
		214c	216c	218c	221c										
SYSDEVS(sysdevs)		158d	617c												
kbdmask															
R804XDVR(a804xdvr)		172c	173c	198c	199c	385c									
HPHIL(hphil)		234c	272s												
SYSDEVS(sysdevs)	***	166d													
TAIL		87c	92c												
kbdpollhook															
R804XDVR(a804xdvr)	***	392c													
HPHIL(hphil)	***	88c													
SYSDEVS(sysdevs)		156d	619c												
kbdreleasehook															
KEYS(keys)		369c	568c												
SYSDEVS(sysdevs)		247d	646c												
kbdrehook															
R804XDVR(a804xdvr)	***	391c													
SYSDEVS(sysdevs)		153d	430c	614c											
kbdsetup															
KEYS(keys)		588c	589c												
SYSDEVS(sysdevs)		371d	428d												
kbdstatus															
R804XDVR(a804xdvr)		114d	119c	120c	121c	132d	134c	135c							
KEYS(keys)		355d	358c	373d	426c	428c	429c	430c							
kbdssysmode															
R804XDVR(a804xdvr)		355c	398c												
KEYS(keys)		450c	462c												
SYSDEVS(sysdevs)		289d	650c												
kbdtranshook															
R804XDVR(a804xdvr)		105c	403c												
SYSDEVS(sysdevs)		287d	651c												
kbdtype															
R804XDVR(a804xdvr)		214c	216c	217c	222c	341c	399c								
HPHIL(hphil)	***	364c													
KEYS(keys)	***	310c	442c												
MOUSE(mouse)	***	181c													
NONUSKBD1(non_us_kbd1)	***	94c	116c	154c	165c										
NONUSKBD2	***	127c													
SYSDEVS(sysdevs)		157d	616c												
kbdwaithook															
KEYS(keys)	***	112c	579c												
M8KSYS(ci)		140c													
SYSDEVS(sysdevs)		246d	64Sc												
kbufptr															
SYSDEVS(sysdevs)		235d	241d												
kbufrec															
SYSDEVS(sysdevs)		236d	237d												
kbufrecptr															
SYSDEVS(sysdevs)		236d	245d												
kbuftype															
SYSDEVS(sysdevs)		234d	235d												
kclear															
KEYS(keys)		87c	411c												
SYSDEVS(sysdevs)		232d	540c												
kcodes															
MOUSE(mouse)		43d	107c												
kdisplay															
DGL_C_OUT(dgi_confg_out)	***	11069c													
KEYS(keys)		457c	469c												
PRINTER(prtdvr)	***	158c													
SYSDEVS(sysdevs)		232d	545c												
keep_protected_directories															
SRM_DRV	***	1974c													
SRM_DRV(srm)	***	589d													
keepworkdirs															
SRMM(srmamodule)		147d	162c	164c	170c	171c									
kernel_initialize															
KERNEL	***	1329c													
KERNEL(general_0)		765d	964d												
key															
R804XDVR(a804xdvr)		326d	346c	348c	350c	352c	356c	357c	358c	359c	360c	361c	362c	363c	365c
KEYS(keys)		162c	164c	165c	166c	167c	168c	169c	170c	171c	172c	173c	174c	175c	176c
		177c	178c	193c	217c	219c	356d	360c	362c	364c	366c	368c	433c	447c	449c
		451c	452c	478c	479c	481c	482c	483c	485c	486c	488c	490c	493c	494c	495c
		508c	509c	524c	527c	528c	529c	544c	548c	553c	566c	568c			
NONUSKBD1(non_us_kbd1)	***	69c													
SYSDEVS(sysdevs)	***	264d													
keyboard															
ETU		24d	29d	202c	298c										
INIT		30d	2516d												
LIBRARIAN		24d	37d	101c	277c	2095c									
M8KSYS	***	28d													
INIT(midecs)	***	47c													
TAPEXUP		8d	956d	979c	1504c										
keyboardtype															
SYSDEVS(sysdevs)		145d	157d												
keybuffer															
CRT(crt)	***	707c													
GCRTR(ctrb)	***	521c													
KEYS(keys)		83c	110c	118c	159c	181c	366c	409c	456c	467c	583c				
PRINTER(prtdvr)		157c	181c	184c											
SYSDEVS(sysdevs)		245d	517c	520c	524c	634c	635c								
keybufferaddr															
CRT(crt)		90d	136d	706c											
GCRTR(ctrb)	***	69d													
SYSDEVS(sysdevs)	***	90d													
keybuffersize															
CRT(crt)		92d	138d												
GCRTR(ctrb)	***	71d													
SYSDEVS(sysdevs)	***	91d													

11-103

<hr/>															
keytrans OUT(dgl_conf_out) *** 11069c															
KEYS(keys) 87c 125c 368c 411c 412c 457c 469c 566c 568c															
PRINTER(prtdvr) 158c															
SYSDEV(S(sysdevs)) 389d 512d															
keyclass															
KEYS(keys) 203c 270c 295c 296c 298c 335c 338c 339c 340c															
NONUSKBD1(non_us_kbd1) 48c 152c 194c 196c															
NONUSKBD2 171c 177c 196c 207c 210c 213c 215c 217c 220c 222c															
SYSDEV(S(sysdevs)) *** 273d															
keyloopup															
NONUSKBD1(non_us_kbd1) 80d 138c 139c															
keys															
KEYS 595d															
KEYS(keys) 32d 217c 272c 274c 276c 285c 289c 303c 307c 317c 324c 325c 329c 330c															
NONUSKBD1(non_us_kbd1) 69c 138c 139c 141c 149c 150c 156c 157c 158c 159c 160c 161c 162c 163c															
NONUSKBD2 168c 169c 184c															
SYSDEV(S(sysdevs)) 139c 140c 141c 142c 147c 148c 160c 162c 167c 187c 194c															
keyservice *** 274d															
KEYS(keys) 373d 502c 580c															
keysinit *** 30d															
KEYS															
keystate															
MOUSE(mouse) 62d 70c 115c															
keystram															
M68KSYS(c1) 64d 319c 330c 438c 540c 541c 542c 545c 862c 1054c															
keytab:e															
KEYS(keys) 201c 273c 275c 277c 278c 291c 295c 296c 304c 307c 317c 320c 321c 335c															
NONUSKBD1(non_us_kbd1) 339c 339c 340c 343c															
NONUSKBD2 46c 138c 139c 141c 149c 150c 152c 156c 157c 158c 159c 160c 161c 162c															
NONUSKBD2 163c 168c 169c 171c 178c 179c 180c 183c 184c 188c 189c 193c 194c 196c															
NONUSKBD2 139c 140c 141c 142c 145c 153c 157c 158c 160c 161c 162c 166c 167c 171c															
NONUSKBD2 175c 176c 177c 181c 182c 183c 186c 187c 193c 194c 196c 199c 207c 210c															
SYSDEV(S(sysdevs)) *** 213c 215c 217c 220c 222c															
keytrans															
A804X(DVR(a804xdvr)) 32Ed 403c															
keytrans type															
SYSDEV(S(sysdevs)) 257d 288d															
keytype															
SYSDEV(S(sysdevs)) 258d 285d 273d															
kgetchar															
KEYS(keys) *** 125c															
SYSDEV(S(sysdevs)) 232d 526c															
kgetlat															
KEYS(keys) *** 412c															
SYSDEV(S(sysdevs)) 233d 558c															
killchar															
INIT(fs) *** 793d															
MOREFSYS(mfs) 67c 78c															
kind															
INIT(fs) 754d 770d															
M68KSYS(c1) 914d 918c															
kmaxbufsize															
SYSDEV(S(sysdevs)) 229d 234d 639c															
kntype															
MOUSE(mouse) 57d 58d															
knob_wait_locator															
DGL_KNOB(dgl_knob_in) 18036d 18174c															
knob_device_rec															
DGL_CONF(dgl_conf_in) *** 12023d															
GLE_KNOB(gle_knob_in) 18009d 18010d															
<hr/>															
knob_device_rec_ptr															
DGL_CONF(dgl_conf_in) 12029d 12042d															
GLE_KNOB(gle_knob_in) 18009d 18083c 18119c 18186c 18197c 18216c															
knob_device_rec_spare															
DGL_CONF(dgl_conf_in) 12023d 12053c															
knob_dummy															
GLE_KNOB(gle_knob_in) 18075d 18225c 18226c															
knob_echo_gcb															
DGL_CONF(dgl_conf_out) 11161d 11276c 11354c															
knob_get_digitalize															
GLE_KNOB(gle_knob_in) 18194d 18223c															
knob_get_input_plp2															
GLE_KNOB(gle_knob_in) 18080d 18227c 18263c															
knob_input_echo															
GLE_KNOB(gle_knob_in) 18207d 18224c															
knob_rec															
DGL_CONF(dgl_conf_in) 12029d 12034c 12042d 12053c 12054c 12055c 12060c															
knob_sample															
GLE_KNOB(gle_knob_in) 18106d 18221c															
knob_sample_locator															
DGL_KNOB(dgl_knob_in) 18024d 18175c															
knob_start_digitalize															
GLE_KNOB(gle_knob_in) 18183d 18222c															
knob_type															
DGL_CONF(dgl_conf_in) *** 12055c															
GLE_KNOB(gle_knob_in) 18011d 18084c 18229c															
knonadvance															
KEYS(keys) *** 566c															
SYSDEV(S(sysdevs)) 232d 531c															
koptype															
SYSDEV(S(sysdevs)) 232d 389d 512d															
kpassextc															
A804X(DVR(a804xdvr)) 330c 335c															
NONUSKBD1(non_us_kbd1) *** 127c															
SYSDEV(S(sysdevs)) *** 257d															
kpassthru															
A804X(DVR(a804xdvr)) 343c 400c															
SYSDEV(S(sysdevs)) 257d 651c															
kputfilter															
SYSDEV(S(sysdevs)) 233d 567c															
krec															
KEYS(keys) 254d 265c 266c 267c 268c 270c 273c 274c 275c 276c 277c 284c 285c															
kshift_extc 289c 291c 298c 299c 300c 303c 304c															
A804X(DVR(a804xdvr)) *** 368c															
KEYS(keys) *** 264c															
NONUSKBD2 *** 136c															
SYSDEV(S(sysdevs)) *** 257d															
kunit															
INIT('s') *** 760d															
kvld															
CTABLE: 1232d 1234c															
INIT('s') *** 760d															
1															
A804X(DVR(a804xdvr)) 232d 235c 236c															
DGL_POLY(dgl_poly) 20059c 20108d															
DGL_RHS(dgl_raster) *** 17172d															
INIT(LAD(loader)) 1224d 1230c 1232c 1240c															
LIFDR(lifmodule) 80c 93c 103c															
M68KSYS(c1) 592c 599c 604c															
1 flag</															

label_included_flag																			
SRM_DRV	***	1772c																	
SRM_DRV(srm)	***	504d																	
ladd_cmd_buf			407d	412c															
AMIGO(csamigo)																			
lag_base			211d	279c	337c														
DISCHP1B(dischp1b)																			
lagbase			57d	115c	221c														
PRINTER(prtdvr)																			
langcode																			
KEYS(keys)		259c	316c																
NONUSKBD1(non_us_kbd1)		120c	128c																
NONUSKBD2		131c	137c																
SYSDEVS(sysdevs)	***	278d																	
langcom																			
KEYS(keys)		162c	164c	185c	166c	167c	168c	169c	170c	171c	172c	173c	174c	175c	176c				
NONUSKBD1(non_us_kbd1)		177c	178c	191c	193c	194c	196c	199c	201c	419c	423c	568c							
SYSDEVS(sysdevs)	***	42c	44c	45c															
langcomrec		284d																	
SYSDEVS(sysdevs)		261d	284d																
langindex																			
KEYS(keys)		160c	201c	208c	209c	210c	434c	487c	555c										
NONUSKBD1(non_us_kbd1)		60c	61c	62c	199c														
NONUSKBD2	***	203c																	
SYSDEVS(sysdevs)		286d	649c																
langitf																			
A804XDVR(a804xdvr)		183d	186d																
langkeyrec																			
KEYS(keys)	***	254d																	
SYSDEVS(sysdevs)		268d	280d																
langnonitf																			
A804XDVR(a804xdvr)		184d	193d																
langptr																			
SYSDEVS(sysdevs)		282d	285d																
langrecord																			
SYSDEVS(sysdevs)		276d	282d																
langtable																			
KEYS(keys)		160c	201c	203c	256c	257c	312c	313c	314c	434c	487c	555c							
NONUSKBD1(non_us_kbd1)		46c	61c	113c	120c	123c	132c	133c	166c										
NONUSKBD2		129c	131c	133c															
SYSDEVS(sysdevs)		285d	648c																
langtype																			
A804XDVR(a804xdvr)		183d	184d																
SYSDEVS(sysdevs)		146d	153d	273d															
largekbd																			
A804XDVR(a804xdvr)		217c	223c																
NONUSKBD1(non_us_kbd1)		116c	165c																
SYSDEVS(sysdevs)	***	145d																	
last																			
INITLOAD(loader)	***	868d																	
LIBRARIAN		193c	1522c	1582c	1625c	1836c	2384c	2388c	2396c	2413c	2793c	2903d	2905c	2906c	2908c				
UCSD_DAM(ucsdmodule)		2909c	2913c																
SRMDAM(srmdammodule)		659d	707c	743c	756c	793c	805c	811c	830c										
UCSD_DAM(ucsdmodule)		185d	195c	200c	205c	220c	228c	229c	233c	570d	586c	596c	600c	616c					
last_access_date																			
SRMDAM(srmdammodule)		436c	609c																
SRM_DRV(srm)	***	283d																	
last_but																			
GLE_KNOB(gle_knob_in)		18018d	18121c	18155c	18157c	18190c	18202c	18203c	18261c										
last_char																			
PRINTER(prtdvr)		145d	234c																

last_harddisc_lun																			
CTABLE	***	1225c	1814c	1813c															
CTABLE(options)		72d																	
last_index																			
DGL_POLY(dgl_poly)		20421d	20642c	20643c	20650c	20661c	20671c	20875c											
last_knob_dx																			
DGL_KNOB(dgl_knob)		18021d	18066c	18140c	18178c														
last_knob_dy																			
DGL_KNOB(dgl_knob)		18022d	18067c	18141c	18179c														
last_knob_ry																			
DGL_KNOB(dgl_knob)		18019d	18030c	18143c	18176c														
last_knob_rz																			
DGL_KNOB(dgl_knob)		18020d	18031c	18143c	18177c														
last_lun																			
CTABLE(ctr)		308d	887d	893c	898c														
last_subpoly																			
DGL_POLY(dgl_poly)		20916d	20957c	20969c															
last_subpolygon																			
DGL_POLY(dgl_poly)		21035d	21041c	21061c	21081d	21087c	21106c	21126d	21131c	21137c	21157d	21162c	21168c						
last_x																			
DGL_HPGLI(dgl_hpqli)		18046d	18066c	18083c	18085c														
DGL_KNOB(dgl_knob)		18046d	18106c	18115c	18117c														
GLE_KNOB(gle_knob_in)		18020d	18159c	18200c	18259c														
last_y																			
DGL_HPGLI(dgl_hpqli)		18046d	18067c	18083c	18086c														
DGL_KNOB(dgl_knob)		18046d	18107c	18115c	18118c														
GLE_KNOB(gle_knob_in)		18021d	18160c	18201c	18260c														
lastbLock																			
INIT(misc)		537d	556c	559c	589c	625c	626c	628c	663c										
lastbLocks																			
LIBRARIAN		2195d	2206c	2207c	2211c	2212c													
lastfand																			
INIT(misc)	***	703c																	
INITLOAD	***	1707c																	
INITLOAD(bootdammodule)		669c	708c																
INITLOAD(loader)	***	837d																	

lastured													
LIFRAM(lifmodule)		679d	719c	774c	796c	809c	817c	855c	860c	869c	887c		
lastvc													
INITLOAD(bootdammodule)	***	564d											
LIFRAM(lifmodule)		65d	619c	942c									
latency_induced													
CS80(cs80)	***	186d											
CS80(cs80dsr)	***	723c											
TAPEBKUP(cs80tdvdr)		605c	837c										
lavailblk													
UCSD_DAM(ucsdmodule)		354d	361c	362c	365c								
lb													
LIBRARIAN		442d	502c	505c									
lc													
LIBRARIAN		3304d	3349c	3350c	3351c	3353c	3518d	3548c	3552c	3556c	3593c	3607c	
lcommand													
HPHIL(hphil)		76d	198c	298c									
lcommandop													
HPHIL(hphil)	***	298c											
SYSDEVS(sysdevs)	***	306d											
lcount													
SRM_DRV(srm)		1302d	1307c	1310c	1312c	1316c	1329c	1337c	1348c				
lcursaddr													
CRT(crt)		280d	284c	285c	286c	287c	288c						
ldr													
CTABLE	***	1186d											
CTABLE(ctr)	***	239d											
INIT	***	2514d											
INIT(ldr)	***	2268d											
LIBRARIAN	***	27d											
MS8SYS(c1)	***	33d											
SEGMENTER(segmenter)	***	18d											
TAIL	***	32d											
le configured													
HPHIL(hphil)	***	314c											
SYSDEVS(sysdevs)	***	294d											
le error													
SYSDEVS(sysdevs)	***	295d											
le loopdown													
HPHIL(hphil)	***	311c											
SYSDEVS(sysdevs)	***	297d											
le timeout													
SYSDEVS(sysdevs)	***	296d											
left													
CRT(crt)		96d	142d										
GCRT(crtb)	***	75d											
SYSDEVS(sysdevs)	***	93d											
leftchar													
CRT(crt)	***	461c											
GCRT(crtb)	***	332c											
INIT(misc)	***	163d											
leftinbuf													
ETU		424d	432c	448c	454c	472c	481c						
leftover													
INITLOAD(loader)		1209d	1320c	1322c	1328c	1330c							
lefttofer													
ETU		85d	440c	441c	443c	444c	464c	1032c	1045c	1050c	1055c	1102c	1114c

len													
CRT(crt)		552d	573c	585c	593c	605c	607c						
CS80(cs80)		355d	556d	587c									
GLE_KNOB(gle_knob_in)	18045d	18054c											
HEART(hpm)	217d	220c	221c	223c	226c	230c	237c	245c	254c	255c			
INITLOAD(loader)	1225d	1236c	1237c	1238c	1239c	1240c	1251d	1270c	1271c	1272c	1273c	1274c	1400d
LIBRARIAN	1455c	1456c	1457c	1174c	1181c	1317d	1330c	1331c	1332c	1655d	1722d	1782c	1783c
TAPEBKUP(cs80tbr)	1797c	1852d	1901c	1902c	1903c	2003d	2051c	2052c	2053c	2073c	2074c	2075c	2076c
UNITIO(utio)	194d	201c											
len hbyte	28d	30d	53d	65c	74d	86c							
SRM_DRV(srm)	***	303d											
len lobyte													
SRM_DRV(srm)	***	302d											
length													
AMIGO(amigodvr)		431d	610c	617c	626c								
CRT(crt)		403d	415c	421c	424c	427c	437c	440c	441c	455c	457c	475c	529c
CS80(cs80)		575d	587c										531c
CS80(cs80dvr)		988d	1095d	1110c	1112c	1116c	1215d	1250c	1258c	1272c	1275c	1278c	1280c
F9885(f9885dvr)		1302c	1311c	1314c	1318c	1392c	1395c						1283c
GCRT(crtb)		41d	110d	284c	286c								1299c
INITLOAD(bootdammodule)		273d	285c	291c	294c	297c	307c	310c	311c	326c	328c	346c	
INITLOAD(loader)		588d	648d	655c	673c								
INITLOAD(mini)		1219d	1230c										
KEYS(keys)		310d	385d	421c	433c	436c	441c	445c	512c				
MS8SYS(c1)		73d	84s	106c	120c	121c	122c						
PRINTER(prtdvr)		732d	737c	738c	739c	740c							
TAPEBKUP(cs80tbr)		41d	48d	305c	307c	309c							
lentitle		30d	179d	201c	223d	236c							
SRMDAM(srmdammodule)		651d	673c	682c	714c	721c	734c	746c	752c	759c	773c	780c	782c
lerror													791c
HPHIL(hphil)		48d	307c										
letter													
AMIGO(amigodvr)		428d	439d	450d	457d	458d	459d	460d	461d	462d	463d	466c	469c
AMIGO(csamigo)	***	480c	484c										470c
CS80(cs80dsr)	***	162c											472c
CS80(cs80dvr)	***	619c											
CTABLE		982d	1027d	1080c									
CTABLE(brstuff)		1193d	1195d	1251c	1274c	1278c	1364c	1394c	1471c	1526c	1535c	1536c	1542c
CTABLE(ctr)		1556c	1558c	1566c	1596c	1598c	1820c						1548c
CTABLE(scanstuff)	***	833d	1003c	1004c	1008c	1009c							1549c
INIT(initunits)	***	280d	309d	310d	312d	412c	482c	496c	525d	542c	560d	564c	595d
INIT(misc)	***	883c											599c
INITLOAD	***	1053d											838c
INITLOAD(sysglobals)	***	2214c											
MIU	***	282c	288c										
TAPEBKUP	***	1688c											
letter_table		155d											
CTABLE(brstuff)		37c	170c	319d	323c	351c							
letter_table_type		1011c	1088c										
CTABLE(brstuff)		989d	1006c										
level													
SRM_DRV(srm)	***	987d	990d										
level_type													
INITLOAD(sysglobals)	***	304d											
lf													
INIT(misc)	***	165d											
KEYS(keys)	***	360c	482c										

[illegible]

1. CTAL E(ctr)	***	408c																	
HEAT(hpm)		71d	122d	142c	148c	150c	156c	158c	159c	161c	167c	169c	170c	184c	185c				
INIT(isr)		205c	226c	236c	246c	252c	253c												
INIT(ldr)		73c	94c	95c	104c	123c	124c	130c	132c										
INITLOAD	***	2346c	2367c	2388c															
INITLOAD(loader)		1722c																	
INITLOAD(sysglols)	***	1003d	1182c	1189c	1202c	1313c	1339c	1362c	1381c	1415c	1450c	1489c	1487c	1488c	1557c				
LIBARIAN		210d																	
M8K SYS(c1)		1558d	1559c	1560c	1661c	1705c	1743c	1955c	2029c	2718c	2980d	3225c	3639c						
SEGENTER(asm)		907c	933c																
linker		137c	270c																
LIBARIAN	***	24d																	
linker date		39d	231c	1961c	3710c														
LIBARIAN																			
linkerrs	***	1404d																	
SRM DRV(srm)																			
linkfiller		321d	329d	1426c															
SRM DRV(srm)																			
linkfiller type		295d	321d	329d															
SRM DRV(srm)																			
linking																			
LIBARIAN		47d	2837c	2954c	2974c	3009c	3124c	3125c	3350c	3543c	3550c	3565c	3627c	3629c	3633c				
		3640c	3643c	3648c	3649c	3653c	3688c	3732c											
linkmdname	***	2496c																	
INIT(ldr)		1056d	1523c	1524c	1525c														
INITLOAD(loader)																			
linklevel																			
INIT(isr)		42d	49d	52d	64d	73c	75c	82d	89c	90c	92c	108c	111d	118c	119c				
		129c	130c																
linkregaddr		39d	46d	61d	69c	79d	100c												
INIT(isr)																			
linkregmask																			
INIT(isr)		40d	47d	62d	70c	80d	101c												
linkregvalue																			
INIT(isr)		41d	48d	63d	71c	81d	102c												
linkx																			
UCSL DAM(ucsdmodule)		325d	334c	336c	338c	342c	344c	345c	353d	357c	358c	361c	373c	398d	400c				
		401c	405d	408c	409c	411c	421c												
list																			
DGL RAS(dgl_raster)		17247d	17263c	17264c	17260c														
ETU		191d	204c	205c															
HEAT(hpm)		194d	202c	203c	205c	206c													
INITLOAD(loader)		1214d	1252d	1265c	1269c														
LIBARIAN		248d	327c	332c	348c	351c	352c	353c	361c	366c	368c	370c	373c	375c	377c				
		383c	1184c	1254c	1285c	1287c	1289c	1306c	1323c	1331c	1351c	1357c	1375c	1395c	2086c				
		2220c	2350c	2497c	2541c	2550c	2615c	2727c	2987c	2988c	2999c	3000c	3018c						
listaddr																			
INIT(ldr)		2392c	2395c																
INITLOAD(loader)		1021d	1188c	1291c	1293c	1294c	1310c	1348c											
LIBARIAN		400c	1354c	1356c	1764c	1787c	1790c												
listdefs																			
LIBARIAN		1315d	1465c																
listerr																			
IOLIS(hpib_2)		1781d	1853d																
listerr constant																			
GLE APIS(gle_hpib_10)		10215c	10303c																
IOLIS(hpib_1)		537c	625c																
IOLIS(hpib_2)	***	1856c																	
KERNEL(iodeclarations)	***	317d																	
listerr																			
IOLIS(hpib_3)		2039d	2135d	2138c															

listexts																			
LIBARIAN		1343d	1466c																
listfilename																			
LIBARIAN		43d	2934c	3527c	3725c	3726c													
listimg																			
INIT		30d	2517d																
LIBARIAN		44d	244c	254c	261c	327c	328c	329c	330c	331c	332c	333c	334c	348c	349c				
		351c	352c	353c	361c	366c	368c	370c	373c	375c	379c	382c	383c	472c	474c				
		1184c	1188c	1189c	1255c	1257c	1261c	1263c	1264c	1285c	1287c	1289c	1306c	1323c	1331c				
		1335c	1351c	1357c	1377c	1385c	2086c	2220c	2351c	2352c	2497c	2500c	2541c	2564c	2575c				
		2615c	2727c	2729c	2730c	2939c	2940c	2941c	2987c	2988c	2999c	3001c	3002c	3003c	3018c				
		3705c	3706c	3726c															
MIUI	***	8d																	
listinstruction																			
LIBARIAN		414d	1308c																
listlen																			
LIBARIAN		1718d	1729c	1735c	1746c	1747c	1748c	1752c	1760c	1761c	1770c	1779c	1792c	1999d	2015c				
		2021c	2032c	2033c	2034c	2037c	2045c	2046c	2058c	2068c	2081c								
listlength																			
INITLOAD(loader)		1174d	1215d	1222d	1242c														
listin																			
LIBARIAN		260d	384c	1183c	1307c	1309c	1325c	1338c	1353c	1360c	1387c	1407c	3004c	3019c	3023c				
listipn																			
INITLOAD(loader)		1221d	1229c	1243c															
listsize																			
INIT(ldr)	***	2392c																	
INITLOAD(loader)		1022d	1188c	1292c	1310c	1349c													
LIBARIAN		400c	1354c	1758c															
listtext																			
LIBARIAN		1365d	1464c																
lite																			
GEN(dgl_gen)		3048d	3049d	3166d	3176c	3177c	3178c	3181c	3182c	3183c	3184c	3185c	3186c	3190d	3219c				
		3221c	3223c	3227c	3228c	3229c	3231c	3237c											
1keytype																			
MIOUS(mouse)		50d	77d																
kind																			
ETU		309d	316c	320c															
limit																			
M8K SYS(c1)		209d	224c																
llo message																			
IOLIS(hpib_2)	***	1891c																	
KERNEL(iodeclarations)	***	305d																	
lmaxde/ices																			
SYSD SYS(sysdevs)		299d	346d	347d															
lms dgl type	***	59d																	
CTAB E(options)																			
lmsr																			
PRINTER(prtdvr)		166d	186c	187c	188c														
lname																			
INITLOAD(bootdmodule)	***	585d																	
lrom																			
EDRIS(edriver)		81c	181c																
EPROIS(eproms)		15d	80c																

II-111

lowherp		2430c	2431c	2478c	2479c											
INIT(ldr)		1049d	1114c	1115c	1143c	1151c	1165c	1229c	1276c	1319c	1321c	1332c	1352c	1428c	1452c	
INITLOAD(loader)		1467c	1471c	1546c	1553c	1598c	1599c	1641c	1642c							
LIBARIAN		127c	128c	1393c	1310c	1340c	1362c	1409c	1418c	1514c	1538c	1597c	1604c	1619c	1643c	
		1645c	1729c	1870c	1884c	1909c	1949c	1985c	2015c	2035c	2114c	2121c	2320c	2321c	2326c	
		2331c	2398c	2405c	2437c	2513c	2523c	2525c	2842c	2747c	2798c	2821c	2825c	2840c	2842c	
		2843c	2849c	2873c	2953c	2975c	2990c	3008c	3094c	3095c	3096c	3111c	3124c	3129c	3692c	
		3721c	3722c													
SEGENTER(asm)		188c	189c	236c	242c	329c										
lowhepo																
LIBARIAN		61d	3692c	3721c												
lowid																
HPHIL(hphll)	***	257c														
MOUSE(mouse)	***	186c														
SYSDEVS(sysdevs)	***	338d														
lowkeys																
NOM SKBD2		39d	139c	140c	141c	142c										
lowarg																
LIBARIAN		307d	1200c	1201c	1251c	1252c	1426c	1433c								
lpcrtitype																
HPHIL(hphll)		55d	132d	159d												
lprompt1																
M8K SYS(ci)		977d	1090c													
lpromt12																
M8K SYS(ci)		973d	1090c													
lptr																
CRT(crt)		158d	509c	513c	514c	519c	520c	524c	530c	532c	537c	706c				
lr																
CTABLE(ctr)		333d	334d	335d	720d	727c	729c	733c	737d	745c	746c	747c	752c	791d	793c	
lrec																
INITLOAD(bootdanmodule)		732d	738d													
ls_map																
DGL HPGL(dgl_hpgl)		17191d	17196c													
DGL_RAS(dgl_rasr)		17052d	17057c													
ls_map_def																
DGL HPGL(dgl_hpgl)		17189d	17191d													
DGL_RAS(dgl_rasr)		17050d	17052d													
ls_patterns																
DGL_VARS(dgl_vars)		1303d	1307d													
lsize																
LIBARIAN		710n	715c	716c	717c	721c	722c	726c								
l1																
LIFDAM(lifmodule)		540d	543c	544c												
ltime																
CLOCK(clock)		51d	53c	54c	101d	103c	104c	111c	115d	117c	118c	123c				
M8K SYS(ci)		151c	158c	159c	164c											
SYSDEVS(sysdevs)		458d	460c													
ltitle																
M8K SYS(ci)		593d	598c	599c	602c	625d	633c	634c	636c	639c	640c	642c				
lun																
CS8C(cs80dsr)		613d	616c	618c												
CTABLE		1213d	1224c	1225c	1268d	1274c	1275c	1278c	1522c	1531c	1540c	1554c	1574c			
CTABLE(ctr)		890d	896c	898c	900c	902c	903c	905c	907c							
MIUI		18d	142c	148c	151c	153c	167c	206c	237c	353c	477c	487c	514c			
TAPEBKUP		1048d	1072c	1078c	1081c	1083c	1085c	1095c	1106d	1121c	1124c	1426d	1433c	1434c	1436c	
		1460d	1467c	1468c	1472c											
lun1																
CTABLE		1213d	1816c	1819c	1820c	1822c										
CTABLE(ctr)		307d	875d	880c												
TAPEBKUP		1003d	1007c													
lun2																
CTABLE		1213d	1813c	1814c	1816c	1818c	1819c	1822c								
CTABLE(ctr)		307d	875d	879c												
TAPEBKUP		1003d	1008c													
lun11																
CTABLE(ctr)		303d	306d	823d	827c	828c	832d	837c	838c	840c	841c	842c	850d	854c	855c	
		868d	870c	871c												
		588d	612c	613c	615c	617c	953d	957c	958c	962c						
lv																
TAPEBKUP(cs80tbr)		189d	198c	199c	200c	201c	202c	203c								
lvheader																
LIFDAM(lifmodule)		42d	197d	251c	352c	528c	1144c									
lx																
GEN(dgl_gen)		3032d	3169d	3176c	3181c	3182c	3183c	3184c	3185c	3186c	3281d	3293c	3296c	3313d	3321c	
lxmax																
GEN(dgl_gen)		3018d	3340d	3353c												
LIB(dgl_lib)		20083d	20880d	20902c	20908c											
lxmin																
GEN(dgl_gen)		3018d	3340d	3352c												
LIB(dgl_lib)		20083d	20880d	20902c	20907c											
ly																
GEN(dgl_gen)		3032d	3169d	3177c	3182c	3184c	3186c	3281d	3295c	3297c	3313d	3323c				
lymax																
GEN(dgl_gen)		3019d	3341d	3355c												
LIB(dgl_lib)		20083d	20880d	20902c	20910c											
lymin																
GEN(dgl_gen)		3019d	3341d	3354c												
LIB(dgl_lib)		20083d	20880d	20902c	20909c											
lz																
GEN(dgl_gen)		3169d	3178c	3181c	3183c	3185c										
m																
MIUI		15d	103d	171c	197c	330c	348c	473c								
m9816a																
DGL TOOLS(dgl_tools)	***	20067c														
GLE_RGL(gle_ras_out)	***	8431c														
GLE_TYPES(gle_types)	***	1254d														
m9826a																
DGL TOOLS(dgl_tools)	***	20063c														
GLE_RGL(gle_ras_out)	***	8438c														
GLE_TYPES(gle_types)	***	1255d														
m9836a																
DGL TOOLS(dgl_tools)		20051c	20057c													
GLE_RGL(gle_ras_out)	***	8445c														
GLE_TYPES(gle_types)	***	1256d														
m9836c																
DGL : OUT(dgl_confg_out)		11218c	11238c													
DGL TOOLS(dgl_tools)	***	20048c														
GLE_RGL(gle_ras_out)	***	8452c														
GLE_TYPES(gle_types)	***	1257d														
m9837a																
DGL : OUT(dgl_confg_out)	***	11255c														
GLE_RGL(gle_ras_out)		8383c	8467c													
GLE_TYPES(gle_types)	***	1259d														
m9862a																
DGL : OUT(dgl_confg_out)	***	11321c														
GLE_RGL(gle_ras_out)	***	8459c														
GLE_TYPES(gle_types)	***	1258d														
m_none																
KEYS(keys)		45d	49d	50d	455c	465c	467c	469c								
PRINTER(prtdvr)		181c	183c													
SYSD:VS(sysdevs)		151d	620c													
m_syn																
KEYS(keys)		49d	50d	453c	470c											
SYSD:VS(sysdevs)	***	151d														

II-112

m_ysysshift			45d	48d	49d	453c	465c	471c															
-KEYS(keys)	***	151d																					
SYSDEV5(sysdevs)	***	151d																					
m_u1	***	151d																					
-SYSDEV5(sysdevs)	***	151d																					
m_u2	***	151d																					
-SYSDEV5(sysdevs)	***	151d																					
m_u3	***	151d																					
-SYSDEV5(sysdevs)	***	151d																					
m_u4	***	151d																					
-SYSDEV5(sysdevs)	***	151d																					
mac_ident			501d	515c																			
AMIGO(amigodvr)			501d	515c																			
maddr			INITLOAD(bootdammodule)	***	573d																		
madr			MINIT(asmr)		1337d	1348d																	
maintenance_track_overflow			CS80(cs80)	***	195d																		
CS80(cs80dsr)	***	732c																					
TAPBCKUP(cs80tbdrv)			614c	846c																			
major_index			DGL_POLY(dgl_poly)		20428d	20610c	20623c	20778c	20779c	20780c	20781c	20783c	20784c	20789c	20790c								
make_initial_directories			MUI		371d	550c																	
make_initial_lif_directory			MUI		386d	510c																	
make_pic_current			LIF(dgl_lib)		20029d	20378d	20942c	20973c	21312c														
makedir			LIBRARIAN		1847d	2992c																	
UCSD_DAM(ucsdmodule)			472d	643c																			
makedirectory			INITLOAD(sysglobals)	***	136d																		
LIFDAM(lifmodule)	***	1155c																					
MBSKSYS(ci)	***	825c																					
MUI	***	533c																					
SRMDAM(srmdammodule)			1657c	1748c																			
UCSD_DAM(ucsdmodule)	***	643c																					
makelist			INITLOAD(loader)		1218d	1265c	1291c																
makernewavr			LIBRARIAN		1473d	1909c	1949c	2077c	2656c														
makepatch			LIBRARIAN		2357d	2699c	2702c																
makepatchspace			LIBRARIAN		3213d	3649c																	
makeramvol			MBSKSYS(ci)		771d	1131c																	
mantissa			MOREFSYS(mfs)		34d	415c	438c	461c															
mantissa_digit			MOREFSYS(mfs)		400d	411c	437c	438c	439c	445c	460c	461c	462c	500c									
mantissadigit			MOREFSYS(mfs)		228d	349c																	
mantissadigitb			MOREFSYS(mfs)		266d	316c																	
map			AMIGO(csamigo)		140d	189d	210c	211c	215c	221d	223c	224c	341d	344c	345c	365d	368c	369c					
MINIT(hminit)			285d	288c	289c	303d	364c	366c	373c	391c	403c	437c	439c	446c	485c	486c							
map_array			GLE_RGL(gle_ras_out)	***	8096d																		
map_blu			GLE_RGL(gle_ras_out)		8018d	8335c	8338c	8352c															
<hr/>																							
map_grn			GLE_RGL(gle_ras_out)		8017d	8334c	8337c	8352c															
map_h			DGL_RAS(dgl_raster)		17080d	17111c	17116c	17117c															
map_l			DGL_RAS(dgl_raster)		17081d	17111c	17121c	17129c	17141c	17151c													
map_red			GLE_RGL(gle_ras_out)		8016d	8333c	8336c	8352c															
map_s			DGL_RAS(dgl_raster)		17082d	17111c	17120c	17133c	17145c	17155c													
map_type			AMIGO(csamigo)		127d	140d	189d	191d	193d	194d	196d	197d	198d	199d	200d	201d	202d	203d					
MINIT(hminit)			221d	341d	365d																		
mapfkind			SRMDAM(srmdammodule)		86d	90c	93c	604c	935c														
mapkanatocrt			CRT(crt)		373d	396c																	
GCR(Crtb)			255d	266c																			
mapromextocrt			CRT(crt)		354d	397c																	
maptocrt			CRT(crt)		350d	364c	367c	369c	384c	386c	387c	389c	390c	485c	509c	530c							
GCR(Crtb)			251d	260c	261c	262c	267c	386c	381c	401c													
maptoioresult			SRM_DRV(srm)		1131d	1467c																	
marginal_data			CS80(cs80)		192d	317d																	
CS80(cs80dsr)	***	729c																					
TAPBCKUP(cs80tbdrv)			611c	843c																			
mark_def			GLE_SHARK(gle_shark)		5028d	5033d																	
marker			GLE_GEN(gle_gen)	***	2108c																		
GLE_HPGL(gle_hppl_out)	***	7537c																					
GLE_RGL(gle_ras_out)	***	8405c																					
GLE_TYPES(gle_types)	***	1050d																					
LIB(dgl_lib)			20051d	20735d																			
MBSKSYS(ci)			942d	946c	947c																		
marker_cell_shift			GLE_SHARK(gle_shark)		5019d	5158c	5159c																
marker_data			GLE_SHARK(gle_shark)		5021d	5024d	5025d																
marker_height			GLE_SHARK(gle_shark)		5130c	5137c	5159c	5191c															
GLE_TYPES(gle_types)	***	1117d																					
marker_index			GLE_SHARK(gle_shark)		5081d	5153c																	
marker_index_def			GLE_SHARK(gle_shark)		5029d	5082d																	
marker_number			LIB(dgl_lib)		20051d	20735d	20740c	20744c															
marker_rend			GLE_SHARK(gle_shark)		5023d	5028d	5033d	5034d	5035d	5036d	5037d	5038d	5039d	5040d	5041d	5042d	5043d	5044d					
					5045d	5046d	5047d	5048d	5049d	5050d	5051d	5052d	5053d	5054d	5055d	5056d	5057d	5058d					
					5059d	5060d	5061d	5062d	5063d	5064d	5065d	5066d	5067d	5068d	5069d	5070d	5071d	5072d					
					5073d	5074d	5075d	5076d	5077d	5078d	5079d												
marker_size			GLE_GEN(gle_gen)	***	2180c																		
GLE_HPGL(gle_hppl_out)	***	7538c																					
GLE_RGL(gle_ras_out)	***	8406c																					
GLE_TYPES(gle_types)	***	1051d																					

II-113

[illegible]


```

maxbytes
BUBBLES(bubble)          ***      111c
maxadd
CS80(cs80)                ***      112d
CS80(cs80dvr)             ***     1073c
MINIT(qminit)             ***      924c
maxcountx
SYSDEV5(sysdevs)          ***      321d
maxcounty
SYSDEV5(sysdevs)          ***      322d
maxcountz
SYSDEV5(sysdevs)          ***      323d
maxdata
A804XDVR(a804xdvr)        65d       82c      128c      148c      387c
maxhadd
CS80(cs80)                ***      113d
CS80(cs80dvr)             ***     1073c
MINIT(qminit)             923c      1192c
maxhead
MINIT(xminit)             1403d     1528c     1531c     1533c     1535c     1556c
maximum
GLE_FILE(gle_file_io)     ***     9032d
GLE_HPGL(gle_hppl_out)    7016d     7081c     7522c
GLE_HPGLI(gle_hppl_in)    18016d    18335c
GLE_HPIB(gle_hpib_io)     10043d    10544c
maximum_polygon_vertices
DGL_HPGL(dgl_hppl)        ***     17267c
DGL_INQ(dgl_inq)          ***     6390c
DGL_RAS(dgl_raster)       ***     17600c
DGL_VARS(dgl_vars)        ***     1102d
maximum_term_count
DISCHPIB(dischpib)        506d      511c      512c      513c      515c
maxkana
CRT(crt)                   41d       47d      389c
maxlevel
INITLOAD(sysglobals)      24d       200d
maxlines
MIUI                        227d      258c      261c
TAPEBKUP                   1114d     1145c     1148c
maxmodules
LIBRARIAN                  75d       2852c     2886c     3181c     3183c     3185c     3188c     3189c     3190c     3562c     3731c
maxparm
MINIT(qminit)             1034d     1047c
maxptr
HEAPT(hpm)                 54d       61d
maxrealisc
DC_DRV(init_dc)           ***       703c
DC_DRV(intdc)             ***       238d
DGL_C_OUT(dgl_confg_out)  ***     11313c
GLE_HPIB(gle_hpib_io)     ***     10496c
KERNEL(declarations)      ***       244d
maxrec
SRMDAM(srmdammodule)      1029d     1050c     1051c     1052c     1055c     1088c     1125c     1136c     1148c
SRM_DRV(srm)              1051d     1738d     1765c
maxrefsize
INITLOAD(loader)          ***     1496d
LIBRARIAN                  2280d     2284c
maxromex
CRT(crt)                   357d      358d
maxsadd
CS80(cs80)                ***      114d
CS80(cs80dvr)             ***     1076c
MINIT(qminit)             925c      926c

```


medium parameters		1535c	1548c	1658s	1668s	1678s	1689s	1707s	1723s	1734s					
CTAB:E		303c	825d	543c	544c	545c	546c	547c	548c	549c	550c	551c	552c	553c	554c
CTAB:E(ctr)		555c	599c												
CTAB:E(scanstuff)	***	1144c													
medium size															
CTAB:E(ctr)		722d	729c	730c	732c	739d	746c	748c	749c	751c					
memavail															
ETU	***	938c													
LIBRARIAN	***	3722c													
IFDAM(lifmodule)	***	399c													
SEGMENTER(asm)	***	188c													
UCSD_DAM(ucsdmodule)	***	582c													
memblock															
HEAPT(hpm)		68d	69d	78d	122d										
membytes															
788K SYS(ci)		785d	806c	807c	808c										
memtoprom															
ETU		603d	1088c												
menu															
LIBRARIAN		3517d	3620c												
menuI															
KEYS(keys)		44d	45d	48d	49d	50d									
menu2															
KEYS(keys)		45d	47d												
menustate															
KEYS(keys)		453c	455c	465c	466c	467c	468c								
PRINTER(prtdvr)		181c	183c												
SYSDEVS(sysdevs)		162d	620c												
menutype															
KEYS(keys)	***	44d													
SYSDEVS(sysdevs)		151d	162d												
me															
LIBRARIAN		1866d	1877c	1879c	1881c										
mergedefs															
LIBRARIAN		1995d	2993c												
mergedexts															
LIBRARIAN		1713d	2991c												
mergetext															
LIBRARIAN		1865d	1934c	1951c											
merging															
LIBRARIAN		2132d	2171c	2201c	2202c	2533c									
mesasurement															
CTAB:E(scanstuff)	***	1088d													
message															
MIUI		23d	25c	26c	27c										
TAPERKUP		98d	989c	990c	991c	998d	998c								
TAPERKUP(cs80tbdvr)		715d	746c	747c											
message_length															
CS80(cs80)	***	143d													
SRM_DRV(srm)		322d	330d	1442c	1484c										
message_sequence															
CS80(cs80)	***	141d													
mfclose															
INIT:0A0(bootdamodule)		684d	723c												
mfopen															
INIT:0A0(bootdamodule)		648d	655c	659c	678c	705c									
mfs															
MOREFSYS	***	680d													
MOREFSYS(mfs)	***	5d													
mgator_c															
GLE_TYPES(gle_tyres)	***	1260d													

mgbase															
INIT:0A0(loader)		1399d	1405c	1412c	1413c										
LIBRARIAN		1653d	1689c	1685c	1686c	1708c									
mgcoller															
AMIGO(amigodvr)		513c	530c	678c	734c	822c	898c	956c							
AMIGO(csamigo)		138d	177d	179c	311c	413c									
microsnc value															
DISCHPT8(dischpb)	***	216d													
middlebytes															
INIT(misc)		538d	586c	600c	601c	605c	606c	640c	654c	667c					
mides															
MINIT(bminit)	***	57d													
MINIT(hminit)	***	248d													
MINIT(midecs)	***	10d													
MINIT(mminit)	***	111d													
MINIT(qminit)	***	878d													
MINIT(xminit)	***	1372c													
MIUI	***	11d													
mif															
CS80(cs80)	***	108d													
MINIT(qminit)	***	913c													
man															
ASCII(asciimodule)		141d	145c	149c	202c	222c	248c								
CLOCK(clock)		115d	120c	121c											
DGL_POLY(dgl_poly)	***	20481c													
GEN(dgl_gen)		3014d	3097d	3102c	3103c										
LIB(dgl_lib)	***	21216c													
MINIT(hminit)		262d	273d												
MINIT(midecs)	***	21d													
MINIT(mminit)	***	124c													
MINIT(qminit)	***	910c													
MIUI		295c	301c	313c											
min3															
GEN(dgl_gen)		3205d	3208c	3210c	3212c	3220c									
min_csvp															
MINIT(hminit)		420d	458c	480c											
min_hd_size															
CTAB:E		1378d	1383c												
min_size															
CTAB:E(options)		168d	173d	175d	177d	183d	185d	187d							
min_vp															
MINIT(hminit)		412d	458c												
minhear															
MINIT(xminit)		1403d	1525c	1531c	1556c										
min3															
CS80(cs80dvr)	***	972d													
F888(f888dvr)	***	37d													
INIT(initunits)	***	2135d													
INIT:0AD(bootdamodule)	***	535d													
INIT:0AD(mini)	***	300d													
MINIT(mminit)	***	111d													
PRINTER(prtdvr)	***	37d													
min3tn_name															
CTAB:E(ctr)		360d	658c												
min3dve															
INIT:0AD(mini)		391d	492c												
minlio															
INIT:0A0(bootdamodule)	***	773c													
INIT:0AD(mini)		309d	384d												
min3ndxx															
LIBRARIAN		1720d	1768c	1774c	2001d	2056c	2062c								
min3															
INIT:0AD(bootdamodule)		628d	632c	636c	642c	702c									

II-116


```

minkana
CRT(crt) 40d 47d 389c
minlevel
INITLOAD(sysglobals) 23d 200d
minrealisc
DC_DRV(init_dc) *** 703c
DC_DRV(intdc) *** 238d
DGL_C_OUT(dgl_cfg_out) *** 11313c
GLE_HP18(gle_hp18_io) *** 10495c
KERNEL(general_0) *** 987c
KERNEL(iodeclarations) *** 243d
minromex
CRT(crt) 358d 358d
minsc
ETU 36d 700c 779c
minspare
MINIT(xmininit) 1403d 1525c 1528c 1531c 1543c 1777c 1787c
mintdata
MINIT(mminit) *** 114d 121d 128c
MIUI 330c
minute
A804XDVR(a804xdvr) 279c 308c
CLOCK(clock) *** 80c 120c
INITLOAD(sysglobals) *** 225d
LIBRARIAN *** 235c
LIFDAM(lifmodule) 124c 556c 588c
M88KSYS(ci) 130c 165c 276c 277c
MIUI 81c 83c
SRMDAM(srmdammodule) 374c 388d
UCSD_DAM(ucsdmodule) 523c 553c
minwidth
MOREFSYS(mfs) 564d 571c 577c 589c 590c
misc
CLOCK(clock) *** 33d
CONVERT(convert_text) *** 27d
CRT(crt) *** 33d
DISCHP18(bkgnd) *** 31d
ETU *** 26d
F9885(f9885dvr) *** 37d
GCRT(crtb) *** 33d
INIT 2514d
INIT(fs) *** 739d
INIT(ldr) *** 2270d
INIT(misc) *** 151d
KEYS(keys) *** 33d
LIBRARIAN *** 27d
LIFDAM(lifmodule) *** 26d
M88KSYS(ci) *** 33d
MIUI *** 11d
MOREFSYS(mfs) *** 10d
NONUSK8D1(non_us_kbd1) *** 31d
NONUSK8D2 *** 30d
PRINTER(prtdvr) *** 37d
SEGMENTER(segmenter) *** 18d
SRHAM(srhammodule) *** 33d
SRMDAM(srmdammodule) *** 34d
TAIL *** 32d
TAPEBKUP *** 953d
TAPEBKUP(cs80tbdrv) *** 305d
UCSD_AM(ucsd_am) *** 29d
UCSD_DAM(ucsdmodule) *** 29d

```

```

miscinfo
CRT(crt) *** 110d
GCRT(crtb) *** 43d
SYSDEVS(sysdevs) *** 105d
miscop
LIBRARIAN 734d 966c
miscotype
LIBRARIAN 731d 735d
mitr
CS80(cs80) *** 96d
miui
MIUI *** 8d
mkymap
MOUSE(mouse) 59d 118c
ml
SRM_DRV(srm) 1478d 1484c
mm
CLOCK(clock) 50d 59c 60c 62c 64c 68c
mmaxint
INITLOAD(loader) *** 822c
INITLOAD(sysglobals) 15d 30d
UCSD_DAM(ucsdmodule) *** 46d
mmint
INITLOAD(sysglobals) 14d 30d
mmint
MINIT(mminit) *** 108d
MIUI *** 12d
mminitialize
MINIT(mminit) *** 116d 143d
MIUI 348c
mmx
LIB(dgl_lib) 20091d 20092d 20246d 20258c 20263d 20281c
mmy
LIB(dgl_lib) 20091d 20092d 20246d 20259c 20263d 20282c
mname
LIBRARIAN 2736d 2752c 2753c 2788c
mnotfound
LIBRARIAN 3320d 3448c 3487c
mnus
MOREFSYS(mfs) 33d 423c
mnv
CTABLE *** 1821c
CTABLE(ctr) 562d 582c 583c 589c 590c
CTABLE(options) 136d 173d 175d 177d 179d 181d 183d 185d 187d
mod_power_of_2
CS80(cs80dvr) 1245c 1302c
modblock
INITLOAD(loader) 1210d 1264c 1288c 1318c 1325c 1330c 1343c
moddescptr
CTABLE(ctr) *** 384d
INIT(ldr) 2325d 2352d 2384d
INITLOAD *** 1713d
INITLOAD(loader) 875d 905d 1003d 1007d 1036d 1059d 1067d 1069d 1071d 1073d 1105d 1109d 1178d 1179d
LIBRARIAN 1196d 1207d 1301d 1358d 1369d 1398d 1420d 1475d 1498d
M88KSYS(ci) 54d 55d 81d 316d 1474d 1482d 1852d 1721d 1849d 2002d 2129d 2362d
SEGMENTER(asm) 874d 915d 253d
moddirptr
INITLOAD(loader) 878d 909d
mode
SRMDAM(srmdammodule) 201d 258c 285c
SRM_DRV(srm) 1050d 1737d 1763c
model
LIB(dgl_lib) 20034d 20399d 20404c 20405c

```


II-118

moveleft	ASCII(ascii module)	***	74c	203c	223c	251c										
	CONVERT(convert_text)		73c													
	CRT(crt)		311c	585c	605c	642c	643c	645c								
	CS80(cs80dvr)		1268c	1283c	1295c	1318c										
	EDRIVER(edriver)	***	199c													
	EPROMS(eproms)	***	68c													
	INIT(misc)		600c	646c	649c	659c	664c	675c	677c							
	INITLOAD(loader)	***	1152c													
	INITLOAD(mini)		441c	445c												
	LIBRARIAN		481c	482c	2403c	2894c	3157c	3241c								
	MOREFSYS(mfs)	***	657c													
	SRM_DRV	***	2104c													
	UCSD_AM(ucsd_am)		83c	98c												
moveright	CRT(crt)		341c	607c												
	INITLOAD(loader)		1159c	1617c												
	LIBRARIAN		2319c	2411c												
movesize	ETU		858d	1050c	1051c	1053c	1055c	1097c	1100c	1102c	1106c	1110c				
	LIFDAM(lifmodule)		387d	442c	443c	444c	445c	447c	448c	450c						
moving	GLE_HPGL(gle_hppl_out)		7021d	7104c	7110c	7339c	7342c									
mp	CS80(cs80dvr)		985d	1018d	1023c											
	CTABLE		1217d	1535c	1537c	1541c	1543c	1548c	1551c	1555c	1557c	1559c	1568c	1569c	1570c	1571c
			1575c	1577c	1658s	1662s	1664s	1668s	1672s	1674s	1678s	1682s	1684s	1689s	1694s	1696s
			1697s	1701s	1703s	1707s	1712s	1714s	1717s	1719s	1723s	1728s	1730s	1734s	1739s	1741s
	CTABLE(ctr)		1746s	1755s	1757s											
			311d	313d	314d	315d	578d	585c	586c	587c	597d	599c	600c	604d	608c	610c
	HEAPT(hpm)		611c	615d	617c	621d	623c									
			68s	102c	104c	112c	148c	150c	156c	158c	159c	167c	169c	170c	184c	185c
	LIBRARIAN		196c	226c	236c	246c	252c									
	SRM_DRV(srm)		316d	323c												
			875d	1216c	1379c	1380c	1381c	1382c	1383c	1385c	1387c	1388c	1390c	1391c	1392c	1428c
			1430c	1431c												
mp_type	CS80(cs80dvr)		976d	985d	996d	1018d										
	CTABLE		1217d	1375d												
	CTABLE(ctr)		285d	309d	311d	313d	314d	315d	525d	527d	528d	529d	530d	531d	532d	533d
			534d	535d	536d	537d	538d	540d	578d	597d	604d	615d	621d			
	CTABLE(scanstuff)		1043d	1057d	1135d											
mphydata	MINIT(mminit)	***	115d	132d	139c											
	MIUI		473c													
mrbase	INITLOAD(loader)		1399d	1405c	1410c	1411c										
	LIBRARIAN		1653d	1669c	1674c	1675c	1683c	1684c	1707c							
mscode	INITLOAD(bootdamodule)		572d	800c												
msg	ETU		100d	104c	105c											
msg_packet_ptr	SRM_DRV(srm)		308d	823d												
msg_packet_type	SRM_DRV(srm)		307d	308d	1389c	1429c										
msgline	ETU		866d	943c	980c	988c	991c									
msize	ETU		859d	1096c	1097c	1098c	1099c	1100c								
mstates	KEYS(keys)		47d	466c												
mstring	ETU		1148d	1151c												
msus	CTABLE		1283d	1298c	1301c	1302c	1308c	1316c	1334c	1344c	1358c	1380c	1392c	1394c	1416c	1419c
	CTABLE(brstuff)		1422c	937d	985d	997c	1003c	1004c	1006c	1007c	1008c	1009c	1011c	1012c	1016c	1017c
	INITLOAD(bootdamodule)		587d	628d	632c											
msus1	CTABLE(ctr)		317d	410d	412c	413c	414c	415c	416c							
msus2	CTABLE(ctr)		317d	410d	412c	413c	414c	415c	416c							
msus_array_size	CTABLE		1196d	1199d	1299c	1440c	1524c									
msus_array_type	CTABLE		1199d	1203d	1204c	1205d	1290d									
msus_type	CTABLE		1193d	1195d	1199d	1206d	1239d	1283d	1356d							
	CTABLE(brstuff)		932d	937d	985d											
	CTABLE(ctr)		277d	298d	317d	410d										
msus_match	CTABLE	***	1298c													
	CTABLE(ctr)		317d	410d	412c											
msustype	INITLOAD(bootdamodule)		569d	577d	587d	628d										
mt	CLOCK(clock)		101d	106c	107c	108c	109c									
mt_type	INITLOAD(bootdamodule)		570d	800c												
multiport	CS80(cs80)	***	82d													
multiunit	CS80(cs80)	***	83d													
multiunit	INITLOAD(bootdamodule)	***	571d													
unknown	DGL_TOOLS(dgl_tools)	***	20053c													
	GLE_RGL(gle_ras_out)	***	8430c													
	GLE_TYPES(gle_types)	***	1253d													
mvb	UCSD_AM(ucsdmodule)		68d	109c	116c	139c	154c	622c	633c	651c	677c					
mv	CTABLE(ctr)		562d	584c	586c											
	CTABLE(options)		135d	173d	175d	177d	179d	181d	183d	185d	187d					
my_address	GLE_HPIB(gle_hpib_io)		10119d	10127c	10215c	10301c										
	TOLIB(hpib_1)		395d	443d	451c	537c	623c									
	TOLIB(hpib_2)	***	1827c													
my_boolean	TOLIB(hpib_0)		1747d	1753c	1754c											
my_byte	TOLIB(hpib_3)		2082d	2087c	2088c											
my_code	KERNEL(io declarations)		655d	663d	667c											
my_desc	DC_DRV(intdc)	***	418c													
	KERNEL(general_0)		1203c	1266c												
	KERNEL(io declarations)		500d	514d	528d	681c										
my_msg	TOLIB(general_3)		1112d	1114c	1121c	1122c	1123c	1124c	1125c	1126c	1127c	1128c	1129c	1130c	1131c	1132c
			1133c	1134c	1135c	1136c	1137c	1138c	1139c	1140c	1141c	1142c	1143c	1144c	1153c	1155c
			1156c	1157c	1158c	1159c	1160c	1161c	1162c	1163c	1164c	1165c	1170c	1172c	1173c	1175c
my_num	TOLIB(general_1)		306d	311c	312c											
my_ptr	KERNEL(io declarations)		673d	680c	681c											

II-119

[illegible]

[illegible]

nlsgr	LIBARIAN		303d	539c	1063c	1415c													
nlspr	HEART(hpm)		53d 257c	61d 265c	122d 267c	123d 282c	142c 289c	145c 307c	151c 312c	162c 315c	224c	234c	241c	242c	249c	251c			
nindx	SRLMAM(srmdammodule)		654d	771c	782c	786c													
nintdata	MINT(hminit)		252d 331c	270d	277c	279c													
nlen	MORI FSYS(mfs)	***		27d	29d	88c													
no	GLE TYPES(gle_types)	***	1263d																
no_capslock	KEYS(keys)		216c	265c	284c	300c	320c												
	NONUSKBD1(non_us_kbd1)		171c 153c	178c 157c	179c 158c	180c 175c	188c 176c	189c 181c	193c 182c	183c	199c								
	SYSDVS(sysdevs)	***	269d																
no_card	KERNEL(general_0)	***	1014c																
	KERNEL(iodeclarations)	***	323d																
	PRINTER(prtdvr)	***	290c																
no_control	KEYS(keys)		214c	267c	278c	299c													
	SYSDVS(sysdevs)	***	271d																
no_dan_name	CTABLE(ctr)		349d	633c															
no_data_found	CS80(cs80)	***	170d																
	CS80(cs80dsr)	***	691c 571c		803c														
	TAPBKUP(cs80tbufr)																		
no_extensions	KEYS(keys)		205c	268c	278c	300c	343c												
	SYSDVS(sysdevs)	***	272d																
no_id	KERNEL(general_0)	***	986c																
	KERNEL(iodeclarations)	***	348d																
no_isc	DC DRV(intdc)	***	533c																
	DTCMPB(dischpio)	***	487c 226c		227c														
	DTPA DRV(int dma)		1270c	1300c	1307c	1320c	1333c	1374c	1378c	1381c	1564c	1569c	1573c	1591c	1596c	1600c			
	IOLTB(general_4)		1620c 1294c	1626c 1299c	1647c 1275c	1662c													
	KERNEL(general_0)		1294c	1299c															
	KERNEL(iodeclarations)		295d 1197c	677c 1275c	1278c	1290c	1317c	1318c	1319c										
	SRLM DRV(srm)																		
no_kbc	R804XDVR(a804xdvr)		186d 146d	187d 617c	188d	189d	225c												
	SYSDVS(sysdevs)																		
no_parity	IOLTB(serial_3)		2750c	2771c															
	KERNEL(iodeclarations)	***	401d																
no_proc	SEGENTER(asm)		102c	106c	116c														
no_shift	KEYS(keys)		215c	266c	299c														
	NONUSKBD1(non_us_kbd1)	***	68c																
	SYSDVS(sysdevs)	***	270d																
no_spare_available	CS80(cs80)	***	167d																
	CS80(cs80dsr)	***	685c 565c		797c														
	TAPBKUP(cs80tbufr)																		
no_tfr	IOLTB(general_4)		1269c	1420c															
	KERNEL(iodeclarations)	***	559d																
	SRLM DRV(srm)		1196c	1343c															
nobreak	CRTR(ctr)	***	111d																
	GCRT(ctrb)	***	44d																
	SYSDVS(sysdevs)	***	69d																
nocapabilities	SRLMAM(srmdammodule)		64d	304c	1068c	1401c													
nochain	M88KSYS(ci)		58d	141c	1037c	1068c	1098c	1171c											
nocrt	DGL C_OUT(dgl_confg_out)		11286c	11296c	11335c	11342c	11346c												
	SYSDVS(sysdevs)		113d	631c															
nodam	INIT(initunits)		2141d	2232d	2241c														
node	INITLOAD(bootdammodule)		783d	788c	789c	790c													
nodeallocate	SRLM DRV(srm)		460d	1694c															
nodeccm	INITLOAD(bootdammodule)		779d	786c															
nodestr	INITLOAD		1764c	1765c	1770c	1771c	1789c												
	INITLOAD(bootdammodule)	***	540d																
	TAIL		68c	75c															
nodevice	CTABLE	***	1193d																
	CTABLE(brstuff)		991d	992d	993d	994d	1004c												
	CTABLE(ctr)	***	261d																
	CTABLE(scanstuff)		1111c	1144c															
nodisc	F9885(f9885dvr)		48d	219c															
noerror	F9885(f9885dvr)		48d	206c															
noerrs	HPHIL(hphil)		63d	143c	169c														
nointholb	INITLOAD(svsglobals)	***	284d																
	KERNEL(general_0)	***	1015c																
noisr	INIT(initunits)		2142d	2167d															
	MINI(fxminit)	***	1520c																
nokbd	SYSDVS(sysdevs)		145d	616c															
nokeyboard	DGL C_IN(dgl_confg_in)	***	12050c																
	INITLOAD(svsglobals)	***	285d																
nomap	CRTR(ctr)		171d	363c	384c	686c	689c												
non_chr	KEYS(keys)		159c	181c															
	SYSDVS(sysdevs)		239d	536c	538c	543c	555c	556c	560c	563c	641c								
non_dominant	GLE_HGL(gle_hpgl_out)	***	8079d																
	GLE_HPGI(gle_hpgi_out)	***	7698c																
	GLE_TYPES(gle_types)	***	1093d																
non_dominant_mode	DGL VARS(dgl_vars)	***	1202d																
non_us_kbd1	NONUSKBD1	***	209d																
	NONUSKBD1(non_us_kbd1)	***	29c																

II-122

non_us_kbd2init		28d										
NONUSKBD2	***											
non_alpha_key		439c	565c									
KEYS(keys)		194c										
NONUSKBD1(ncn_us_kbd1)	***	171c	210c	215c	222c							
NONUSKBD2		258d										
SYSDEVS(sysdevs)	***											
nonabortive_iorresult_set		120d	209c									
MIUI		1051d	1098c									
TAPEBKUP												
nonadv_key		194c	335c	438c	565c							
KEYS(keys)		196c										
NONUSKBD1(ncn_us_kbd1)	***	196c	207c	213c	217c	220c						
NONUSKBD2		258d										
SYSDEVS(sysdevs)	***											
nonadvkeys		142d	196c	221c								
KEYS(keys)												
none		3294d	3345c	3530c	3538c	3568c	3587c	3602c				
LIBRARIAN		201d										
SRMDAM(srmddammodule)	***											
nonusinit		27d										
NONUSKBD1	***											
nonzero		402d	414c	430c	449c	459c	498c					
MOREFSYS(mfs)												
nop		542d	541c									
CS80(cs80)		830d	830c									
MIUI(cs80ir)		115d	123c	192d	199c	224d	237c					
TAPEBKUP(cs80tbr)												
nop1		570d	581c									
CS80(cs80)												
nop2		573d	585c									
CS80(cs80)												
nonpassyet		1530d	1538c	1563c	1569c	1571c						
SRMDAM(srmddammodule)												
nonpatch		2368d	2419c	2475c								
LIBRARIAN												
nonpower		48d	216c									
F9885(f9885dvr)												
norange		306d	1199c	1250c	1460c	1461c						
LIBRARIAN												
nonreconfig		62d	144c	170c								
HPHIL(hphil)												
nonrecord		48d	230c	235c								
F9885(f9885dvr)												
nonres		143d	194c									
LIBRARIAN												
normal_completion												
AMIGO(csamigo)	***	69d										
normal_vertex		20386d	20503c	20504c	20726c	20736c						
DGL_POLY(dgl_poly)												
normalized_cos		20320d	20618c	20621c	20624c	20629c	20835c	20836c				
DGL_POLY(dgl_poly)												
normalized_one		20043d	20611c	20624c	20630c	20900c	20901c	20994c	20995c			
DGL_POLY(dgl_poly)												
normalized_sin		20320d	20611c	20618c	20621c	20629c	20630c	20834c	20835c			
DGL_POLY(dgl_poly)												
normshft		156d	270c	295c								
HPHIL(hphil)												
normshftop												
HPHIL(hphil)	***	295c										
SYSDEVS(sysdevs)	***	306d										
norwegian_kbd												
A804XDVR(a804xdvr)	***	191d	206c	209c								
NONUSKBD2		179c										
SYSDEVS(sysdevs)	***	147d										
not_ready												
CS80(cs80)	***	168d										
CS80(cs80dsr)	***	687c										
TAPEBKUP(cs80tdvr)	***	567c	799c									
notbusy												
GLE_RGL(gle_ras_out)	***	8117c										
GLE_TYPES(gle_types)	***	1246d										
notcancels												
LIBRARIAN		1489d	1498c	1505c	1511c							
notdone												
LIBRARIAN		1488d	1498c	1499c	1507c							
notendofarms												
M68KSYS(c1)		409d	436c	447c	469c	471c						
notice												
INITLOAD(loader)	***	945d										
LIBRARIAN		333c	334c	1964c								
notrack												
F9885(f9885dvr)		49d	227c									
notready												
F9885(f9885dvr)		62d	208c									
notype												
LIBRARIAN		298d	1440c									
nounit												
INIT(initunits)		2139d	2228d	2241c								
nowopen												
LIFDAM(lifmodule)		1105d	1130c	1131c	1132c							
nprompt	</td>											

[illegible]

[illegible]

openfileptr																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
-------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

[illegible]

pad2	DGL(DVR(a804xdvr))	***	53d													
	BST(rat)	***	41d													
pad3	HPHIL(hphil)	***	61d													
pad5	HPHIL(hphil)	***	59d													
pad6	HPHIL(hphil)	***	59d													
pad7lo	INIT_OAD(sysglobals)	***	290d													
padding	HPHIL(hphil)	***	49d													
pageblocks	LIBRARIAN		31d	1366d	1394c											
pagebuffer	CONVERT(convert_text)		35d	36d	40d	64c	73c	78c	82c	87d	96c	103c	107c	115c	117c	
pagetype	CONVRT(convert_text)		33d	35d	36d	40d	87d	89d								
pageselect	LIBRARIAN		241d	250c	325c	2937c	3701c									
pagelines	LIBRARIAN		30d	250c												
pagenum	LIBRARIAN		45d	236c	253c	2938c	2942c	3704c	3714c							
pagept	CONVRT(convert_text)		89d	117c												
pages	LIBRARIAN		1369d	1388c	1389c											
pagesize	CONVRT(convert_text)		31d	33d	60c	82c	100c	106c	114c	121c						
	LIBRARIAN		1366d	1382c	1388c	1391c	1392c	1399c	1403c							
	UCSD_AM(ucsd_am)		43d	52c	106c	218c	219c	237c								
pallette	DGL_INQ(dgl_inq)	***	6324c													
	GLE_HPGL(gle_hppl_out)	***	7622c	7629c	7638c	7654c	7664c	7668c	7670c	7686c						
	GLE_TYPES(gle_types)	***	1101d													
paoc	SRRDAM(srmdammodule)		97d	102c	103c	106c	111d	113c	115c	664d	678c					
paocbits	SRRDAM(srmdammodule)		97d	404c	432c	541c	602c	728c	857c	860c						
parameter	CTABIE(ctr)		374d	376c												
	MINIT(cs80ir)		605d	615c												
parameter_bounds	CS80(cs80)	***	139d													
	CS80(cs80dsr)	***	673c													
	TAPEEKUP(cs80tbdvr)		553c	785c												
parameter_field_owner	CS80(cs80dsr)		641d	654c	662c	754c										
	TAPEEKUP(cs80tbdvr)		521d	534c	542c	635c	644c	701d	762c	770c	867c	877c				
parindex	H6BKSYS(c1)		389d	452c	498c											
parity mode	TOLIE(serial_3)		2513d	2737d	2748c	2769c										
parm	MINIT(qminit)		1034d	1041c	1049c											
parm1	DGL_HPGL(dgl_hppl)	***	17213d													
	DGL_INQ(dgl_inq)		6018d	6035d	6051c	6056c										
	DGL_FAS(dgl_faste)		17198d	17210c	17215c	17223c	17251d	17263c	17267c	17268c	17273c					
	LIB(dgl_lib)		20087d	20409d	20421c	20424c										
parm2	DGL_HPGL(dgl_hppl)	***	17214d													
	DGL_INQ(dgl_inq)		6018d	6036d	6052c	6056c										
	DGL_FAS(dgl_faste)		17198d	17210c	17216c	17224c	17252d	17264c	17267c	17269c	17274c					
	LIB(dgl_lib)		20088d	20410d	20422c	20424c										
parm3	DGL_HPGL(dgl_hppl)	***	17215d													
	DGL_INQ(dgl_inq)		6020d	6037d	6053c	6056c	17253d	17265c	17267c	17270c	17275c					
	DGL_FAS(dgl_faste)		17200d	17210c	17217c	17225c										
	LIB(dgl_lib)		20089d	20411d	20423c	20424c										
parptr	H6BKSYS(c1)		393d	433c	455c	456c	499c									
parseoptparm	SRRDAM(srmdammodule)		194c	976c	1060c											
part	DGL_HPGL(dgl_hppl)		17248d	17290c												
	DGL_FAS(dgl_faste)		17552d	17609c												
	ETU		491d	498c	499c	501c	502c	504c	507c	509c	511c	513c	517c	518c	521c	524c
			526c	534d	543c	549c	550c									
	GLE_HPGL(gle_hppl_in)		18231d	18268c	18271c											
partial_length	CS80(cs80dvr)		1220d	1257c	1258c	1268c	1269c	1270c	1272c	1295c	1296c	1297c	1299c	1302c	1303c	1307c
			1308c	1309c	1311c											
partitioning_parameters	CTABIE(ctr)		1536c	1549c	1566c	1820c										
			310d	560d	565c	566c	567c	568c	569c	570c	571c	572c	573c			
pass	ETU		88d	570c	1019c	1030c	1066c	1116c	1118c							
	SRRDAM(srmdammodule)		201d	253c	268c	299c										
	TAPEEKUP		1225d	1343c	1374c	1407c										
pass_control	TOLIE(hpib_2)		1786d	1896d												
pass_rqd	DGL_FOLY(dgl_poly)		20058d	20067c	20071c	20107d										
passarray	SRRDAM(srmdammodule)		50d	53d	75d											
passentry	INIT(misc)	***	201d													
	SRRDAM(srmdammodule)		50d	54d	55d	56d	57d	58d	59d	60d	61d	459d	502d			
passfailed	ETU		532d	573c	582c	583c	595c	596c								
passler	SRRDAM(srmdammodule)		1525d	1533c	1559c	1566c	1593c									
passleng	INITLAD(sysglobals)	***	17d	44d												
	SRRDAM(srmdammodule)	***	1671c													
passtype	ETU		43d	532d												
	INIT(misc)	***	203d													
	INITLAD(sysglobals)	***	44d	101d												
	SRRDAM(srmdammodule)	***	1632d													
password	F9885(f9885dvr)		70d	124c	146c	163c										
	SRRDAM(srmdammodule)		128c	150c	155c	160c	189c	303c	484c	541c	768c	778c	859c	860c	1003c	1067c
			1085c	1094c	1122c	1142c	1145c	1158c	1163c	1192c	1301c	1397c	1400c	1419c		
	SRM_DRV(srm)		291d	316d	410d	440d	605d	1671c								
password_index	SRM_DRV(srm)		398d	1605c												
password_info	SRRDAM(srmdammodule)	***	539c													
	SRM_DRV(srm)	***	409c													
passwordarrayptr	SRRDAM(srmdammodule)		75d	525c	1620c											

II-130

penuip	GLE_MARK(gle_mark)	5028d	5033d	5034d	5035d	5036d	5037d	5038d	5039d	5040d	5041d	5042c	5043d	5044d	5045d
		5046d	5047d	5048d	5049d	5050d	5051d	5052d	5053d	5054d	5055d	5056c	5057d	5058d	5059d
		5060d	5061d	5062d	5063d	5064d	5065d	5066d	5067d	5068d	5069d	5070c	5071d	5072d	5073d
		5074d	5075d	5076d	5077d	5078d	5079d	5162c							
peof	UCSD_DAM(ucsdmodule)	355d	365c	366c	368c										
periodict	A804XDVR(a804xdvr)	***	271c	298c	319c										
	SYSDVS5(sysdevs)		213d												
permanent	INIT(ldr)	2274d	2406d	2481c											
	M68KSYS(c1)	912d	917c	936c	937c	987d	990c	1001c							
permissible	INIT(initunits)	***	2177c												
	INIT(isr)		92c	108c	129c	130c									
	INIT(OAR(sysglobals))	***	275d												
	M68KSYS(c1)	***	296c												
permisrlin	A804XDVR(a804xdvr)	***	386c												
	BUBBLE(bubble)	***	166c												
	DC_DRV(init_dc)	***	655c												
	DJ_DRV(init_discint)	***	260c												
	DMA_DRV(init_dma)	***	268c	272c											
	G_DRV(init_gpio)	***	275c												
	H_DRV(init_hpio)	***	283c												
	INIT(isr)		45d	78d											
	RS_DRV(init_rs)	***	263c												
pfname	INIT(fs)	***	753d												
phile	MOREFSYS(mfs)		30d	64c	75c	526c									
phipor	MINIT(asmr)	1332d	1336d	1343d	1347d										
php	INIT(ldr)	2379c	2427c												
	INIT(OAR(loader))	904d	1145c	1154c	1263c	1288c	1325c	1329c	1387c	1538c	1543c	1595c			
	LIBRARIAN	2963c	3736c												
	SEGMENTER(asm)	***	184c												
phydata	MINIT(hminit)	286d	289c	295c											
	MINIT(mminit)	134d	139c												
	MUI	403d	438d	473c	474c	475c	476c	477c							
phyle	INIT(OAR(loader))	874d	937d												
	LIBRARIAN	***	68d												
physical_cylm	INIT(Cs80ir)	***	532d												
physical_data	INIT(dminit)	60d	95d	97d											
	MINIT(hminit)	253d	283d	286d											
	MINIT(midecs)	***	26d												
	MINIT(mminit)	115d	132d	134d											
	MINIT(qminit)	883c	919d												
	MUI	403d	438d												
physical_head	INIT(Cs80ir)	***	533d												
physical_sect	INIT(Cs80ir)	***	534d												
physical_sector_size	SRM_DRV(srm)	***	613d												
physical_size	SRMDAM(srmdamodule)	***	805c												
	SRM_DRV(srm)	***	286d												

pindex	CRT(crt)	***	231d												
pindex	SRMDAM(srmdamodule)	653d	673c	678c	679c	725c	769c	771c	777c	786c					
pio_imp_ptr	BUBBLE(bubble)	***	126d												
	DC_DRV(intdc)	***	411d												
	F988S(f988Sdvr)	***	81d												
	KERNEL(ioeclarations)	535d	540d	571d	673d										
pisrib	A804XDVR(a804xdvr)		89d	115d											
	BUBBLE(bubble)		32d	124d	144d										
	DC_DRV(extdc)	***	199d												
	DC_DRV(intdc)		245d	305d											
	DJ_DRV(extdi)	***	156d												
	DMA_DRV(init_dma)		169d	170d	182d	199d									
	G_DRV(extg)	***	173d												
	H_DRV(exth)	***	171d												
	INIT(initunits)	2142d	2167d												
	INIT(isr)		43d	50d	53d	56d	65d	83d	85d	86d	112d	114d	115d	137d	
	INIT(OAR(sysglobals))	201d	202d	210d	213d										
	RS_DRV(rs)	***	173d												
pk_crt	SRM_DRV(srm)	874d	993d												
pl	ETU	295d	297c												
	M68KSYS(c1)	69d	563c	984d	1090c	1092c	1094c								
plane1_addr	GLE_RGL(gle_ras_out)	***	8034d												
plane1_offset	GLE_RGL(gle_ras_out)	***	8035d												
plane2_offset	GLE_RGL(gle_ras_out)	***	8036d												
plane3_offset	GLE_RGL(gle_ras_out)	***	8037d												
plfirst	M68KSYS(c1)	985d	1062c	1090c	1092c	1111c									
plinfo	INIT(isr)	86d	90c	94c	107c	115d	119c	123c	132c						
pls	MOREFSYS(mfs)	33d	412c												
pm8845addrreg	CRT(crt)	169d	265c	676c											
pm8845cmdreg	CRT(crt)	170d	266c	677c											
pmmod	LIBRARIAN	3214d	3221c	3222c	3225c										
pnsa	SRM_DRV	1782d	1787d	1795c	1796c	1904d	1913c	1967d	1974c						
	SRM_DRV(srm)	928d	1007d	1015d	1023d	1043d	1057d	1062d	1079d	1091d	1549d	1558c	1587d	1596c	1625d
		1634c	1635c	1730d	1745c	1747c									
point	ETU	490d	506c	511c											
	LIBRARIAN	35d	196c	1644c	1903c	1985c	2118c	2237c	2274c	2297c	2299c	2319c	2403c	2411c	
point_array	DGL_PILV(dgl_poly)	20396d	20409d												
point_count	DGL_PILV(dgl_poly)	20926d	20956c	20964c	20967c	20971c	20982c								
point_def	DGL_PILV(dgl_poly)	20394d	20401d	20402d											
point_dfl	DGL_PILV(dgl_poly)	20393d	20396d												
point_ready	GLE_HHGL(gle_hpgl_ir)	18241d	18266c												

II-132

pointer																	
HEAPT(hpm)	61d	66d	71d	75d	86c	93d	99d	110d	135c	193d	210d	213d	216d	286d			
INITLOAD(loader)	305d																
SRMAM(srmammodule)	1098d	1330c	1490c	1563c													
pointertrick	47d																
HEAPT(hpm)	62d	87d	94d														
poliblock	39d	64d															
MOUSE(mouse)	50d	324c	336c														
polling	50d	324c	336c														
HPHIL(hphil)	50d	324c	336c														
pollisr	113d	148c	397c														
A804XDVR(a804xdvr)	113d	148c	397c														
poly_entry_def	17223d	17227d	17228d	17229d	17230d	17231c	17232d	17233d	17234d	17235d	17236d	17237d	17238d	17239d			
DGL_HPGL(dgl_hppl)	17240d	17241d	17242d														
DGL_RAS(dgl_raster)	17547d	17562d	17563d	17564d	17565d	17566d	17567d	17568d	17569d	17570d	17571d	17572d	17573d	17574d			
DGL_VARS(dgl_vars)	17575d	17576d	17577d														
poly_table_def	1032d	1038d															
DGL_VARS(dgl_vars)	1038d	1039d	1343d														
poly_table_def_space																	
DGL_HPGL(dgl_hppl)	***	17283c															
DGL_RAS(dgl_raster)	***	17653c															
DGL_VARS(dgl_vars)	***	1343d															
poly_table_ptr																	
DGL_HPGL(dgl_hppl)	***	17283c	17285c														
DGL_INQ(dgl_inq)	***	6074c															
DGL_POLY(dgl_poly)	***	20153c	20182c														
DGL_RAS(dgl_raster)	***	17653c	17655c														
DGL_VARS(dgl_vars)	***	1116d															
poly_table_ptr_def																	
DGL_VARS(dgl_vars)	***	1039d	1116d														
poly_table_size																	
DGL_HPGL(dgl_hppl)	***	17284c															
DGL_RAS(dgl_raster)	***	17652c	17654c														
DGL_VARS(dgl_vars)	***	1006d	1038d														
polygon																	
DGL_POLY(dgl_poly)	***	20029d	21147d														
GLE_GEN(gle_gen)	***	2188c															
GLE_HPGL(gle_hppl_out)	***	7551c															
GLE_RGL(gle_ras_out)	***	8419c															
GLE_TYPES(gle_types)	***	1055d															
polygon_dev_dep																	
DGL_POLY(dgl_poly)	***	20024d	21071d														
polygon_fill_factor																	
DGL_POLY(dgl_poly)	***	20895c	21004c														
GLE_HPGL(gle_hppl_out)	***	7703c															
GLE_TYPES(gle_types)	***	1098d															
polygon_simulation																	
DGL_POLY(dgl_poly)	***	20195d	20210c	20243c													
polygon_solid_fill																	
DGL_POLY(dgl_poly)	***	20892c	20988c														
GLE_HPGL(gle_hppl_out)	***	7704c															
GLE_RGL(gle_ras_out)	***	8305c															
GLE_TYPES(gle_types)	***	1099d															
polygon_support																	
DGL_INQ(dgl_inq)	***	6402c															
DGL_POLY(dgl_poly)	***	21046c	21091c														
GLE_HPGL(gle_hppl_out)	***	7687c															
GLE_TYPES(gle_types)	***	1096d															
polyline																	
lib(dgl_lib)	20047d	20692d															

porient																	
DGL_INQ(dgl_inq)	6024d	6062d	6077c														
DGL_POLY(dgl_poly)	20017d	20167d	20181c	20185c													
pos_absolute	***	238d															
SRM_DRV(srm)	***	239d															
pos_sec	***	239d															
SRM_DRV(srm)	***	239d															
posit																	
ASCII(ascii module)	43d	49c	55d	59c	60c	200c	256c	260c	275c	276c	290c	294c	310d	322c			
EDRIVER(edrive)	40d	78d	115c	116c	117c												
position_type	***	1943d	680d	1087d													
SRM_DRV(srm)	***	237d															
pospack	***	1941d															
SRM_DRV(srm)	***	1085d															
possible_bytes_transferred																	
AMIGO(amigodvr)	797d	866c	867c	868c													
possibly_correctable_data_error																	
AMIGO(amigodvr)	812c	834c	838c	872c													
AMIGO(csamigo)	***	76d															
power_fail	***	162d															
CS80(cs80)	***	682c	699c														
CS80(cs80dsr)	***	562c	579c	794c	811c												
TAPEBKUP(cs80tdvr)	***	562c	579c	794c	811c												
powerup	***	1779c															
INITLOAD	***	1779c															
pp																	
CTABLE	1218d	1536c	1537c	1549c	1551c	1566c	1571c	1820c	1821c								
CTABLE(ctr)	311d	578d	582c	583c	584c	586c	589c	590c									
pp_type	***	1218d															
CTABLE	***	310d	311d	560d	562d	578d	181d	183d	185d	187d							
CTABLE(ctr)	***	133d	173d	175d	177d	179d											
CTABLE(options)	***	753d															
ppath	***	753d															
INIT(fs)	***	753d															
ppc_message																	
IOLIB(hpib_2)	***	1927c	1941c														
KERNEL(iodeclarations)	***	302d															
ppd_message	***	1942c															
IOLIB(hpib_2)	***	313d															
KERNEL(iodeclarations)	***	313d															
ppe_message	***	1928c															
IOLIB(hpib_2)	***	312d															
KERNEL(iodeclarations)	***	312d															
ppoll																	
IOLIB(hpib_3)	2031d	2080d	2088c														
ppoll_configure	1788d	1915d															
IOLIB(hpib_2)	1791d	1934d															
ppoll_unconfigure	1791d	1934d															
IOLIB(hpib_2)	1791d	1934d															
pprb	258d	263c	264c	423d	428c	429c											
DISCHPIB(dischoib)	258d	263c	264c	423d	428c	429c											
ppsa	931d	1028d	1048d	1630d	1635c	1735d	1747c										
SRM_DRV(srm)	931d	1028d	1048d	1630d	1635c	1735d	1747c										
ppu_message	***	1945c															
IOLIB(hpib_2)	***	307d															
KERNEL(iodeclarations)	***	307d															
predecr																	
LIBRARIAN	733d	740d	786c	900c													
prefix																	
CRT(crt)	105d	151d															
GCRT(crtb)	***	84d															
SYSDEVS(sysdevs)	***	98d															

[illegible]

proc_output_esc																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											</
-----------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	----

ptrrclose		
SRM_DRV(srm)	835d	887d
ptrrcopy		
SRM_DRV(srm)	837d	889d
ptrrcratefile		
SRM_DRV(srm)	839d	891d
ptrrcratelink		
SRM_DRV(srm)	841d	893d
ptrrexchange		
SRM_DRV(srm)	843d	895d
ptrrfileinfo		
SRM_DRV(srm)	845d	897d
ptrrangclean		
SRM_DRV(srm)	847d	899d
ptrrhead		
SRM_DRV(srm)	872d	923d
ptrrinit		
SRM_DRV(srm)	849d	901d
ptrrlock		
SRM_DRV(srm)	851d	903d
ptrropen		
SRM_DRV(srm)	853d	905d
ptrrpos		
SRM_DRV(srm)	855d	907d
ptrrpurge		
SRM_DRV(srm)	857d	909d
ptrrread		
SRM_DRV(srm)	859d	911d
ptrrsetdate		
SRM_DRV(srm)	861d	913d
ptrrseteof		
SRM_DRV(srm)	863d	915d
ptrrunlock		
SRM_DRV(srm)	865d	917d
ptrrvol		
SRM_DRV(srm)	867d	919d
ptrrwfile		
SRM_DRV(srm)	869d	921d
ptrscaleyoualive		
SRM_DRV(srm)	824d	876d
ptrscal		
SRM_DRV(srm)	826d	878d
ptrscalpass		
SRM_DRV(srm)	828d	880d
ptrschangeprotect		
SRM_DRV(srm)	830d	882d
ptrschangevolume		
SRM_DRV(srm)	832d	884d
ptrsclose		
SRM_DRV(srm)	834d	886d
ptrscopy		
SRM_DRV(srm)	836d	888d
ptrscratefile		
SRM_DRV(srm)	838d	890d
ptrscratelink		
SRM_DRV(srm)	840d	892d
ptrsexchange		
SRM_DRV(srm)	842d	894d
ptrrfileinfo		
SRM_DRV(srm)	844d	896d
ptrsrgclean		
SRM_DRV(srm)	846d	898d
ptrrhead		
SRM_DRV(srm)	871d	922d

ptrsinit		
SRM_DRV(srm)	848d	900d
ptrslock		
SRM_DRV(srm)	850d	902d
ptrsopen		
SRM_DRV(srm)	852d	904d
ptrspos		
SRM_DRV(srm)	854d	906d
ptrspurge		
SRM_DRV(srm)	856d	908d
ptrsread		
SRM_DRV(srm)	858d	910d
ptrssetdate		
SRM_DRV(srm)	860d	912d
ptrsseteof		
SRM_DRV(srm)	862d	914d
ptrsunlock		
SRM_DRV(srm)	864d	916d
ptrsvol		
SRM_DRV(srm)	866d	918d
ptrswfile		
SRM_DRV(srm)	868d	920d
ptrtable		
INITLOAD(loader)	880d	1045d
ptrtablepr		
INITLOAD(loader)	880d	913d
purge		
INIT(fs)	***	744d
purge_cld_link		
SRM_DRV	***	1817c
SRM_DRV(srm)	***	522d
purge		
LIFDAH(lifmodule)	662d	671c 998c 1131c
purgefile		
INIT(misc)	***	434c
INITLOAD(sysglobals)	***	135d
LIFDAH(lifmodule)	***	1131c
SRMDAH(srmdanmodule)	1664c	1783c
UCSD_DAH(ucsdmodule)	***	638c
purgeit		
UCSD_DAH(ucsdmodule)	397d	638c
purgename		
INITLOAD(sysglobals)	***	134d
LIFDAH(lifmodule)	***	1149c
SRMDAH(srmdanmodule)	1661c	1768c
UCSD_DAH(ucsdmodule)	***	639c
purgeolj		
SRM_DRV	1791d	1817c
SRM_DRV(srm)	***	1086d
UCSD_DAH(ucsdmodule)	428d	639c
purgepack		
SRMDAH(srmdanmodule)	1143c	1257c 1302c 1367c
SRM_DRV	***	1965d
SRM_DRV(srm)	***	1089d
putbuffer		
UCSD_AM(ucsd_am)	68d	108c 109c 119c 143c 205c
putchar		
UCSD_AM(ucsd_am)	113d	127c 129c
putenviron		
UCSD_AM(ucsd_am)	104d	174c 185c
putmenu		
ETU	***	1148d
putref		
LIBRARIAN	2308d	2589c 2703c

[illegible]

[illegible]

[illegible]

reg changeprotect	SRM_DRV(srm)	139d	1639c	1654c
reg close	SRM_DRV(srm)	131d	1688c	1693c
reg copy	SRM_DRV(srm)	150d	1418c	1713c 1724c
reg create	SRM_DRV(srm)	***	1776c	
	SRM_DRV(srm)		137d	1419c 1751c
reg createlink	SRM_DRV(srm)	1800c	1821c	
	SRM_DRV(srm)	***	138d	
reg flock	SRM_DRV(srm)	1890c	1898c	
	SRM_DRV(srm)		128d	1416c
reg funlock	SRM_DRV(srm)	2052c	2060c	
	SRM_DRV(srm)	***	129d	
reg gang_cleanup	SRM_DRV(srm)	***	1871c	
	SRM_DRV(srm)	***	153d	
reg info	SRM_DRV(srm)	1852c	1860c	
	SRM_DRV(srm)	***	130d	
reg init_vol	SRM_DRV(srm)	***	145d	
reg label	SRM_DRV(srm)	146d	1667c	1677c
reg log_addr	AMIGO(csamigo)	49d	412c	
reg old_read	SRM_DRV(srm)	***	121d	
reg open	SRM_DRV(srm)	1917c	1937c	
	SRM_DRV(srm)	***	134d	
reg position	SRM_DRV(srm)	1950c	1961c	
	SRM_DRV(srm)	***	123d	
reg purgelink	SRM_DRV(srm)	1978c	1991c	
	SRM_DRV(srm)	***	135d	
reg read	SRMAM(srmamodule)	78c	85c	89c
	SRM_DRV(srm)	***	2006c	
	SRM_DRV(srm)		124d	1433c 1443c
reg set_eof	SRM_DRV(srm)	2031c	2041c	
	SRM_DRV(srm)	***	125d	
reg status	AMIGO(csamigo)	47d	310c	
reg syndrome	AMIGO(csamigo)	48d	334c	
reg volstatus	SRM_DRV(srm)	2070c	2077c	
	SRM_DRV(srm)	***	144d	
reg write	SRMAM(srmamodule)	125c	132c	136c
	SRM_DRV(srm)	***	2092c	
	SRM_DRV(srm)	***	122d	
reg xchg_data	SRM_DRV(srm)	***	141d	
reg xchg_open	SRM_DRV(srm)	1833c	1841c	
	SRM_DRV(srm)	***	149d	

[illegible]

[illegible]

[illegible]

[illegible]

sc	BUBBLES(bubble)		63c	64c	67c	143d	149c	150c									
	CS80(cs80dsr)	***	620c														
	CTABLE		1191d	1243d	1245c	1274c	1278c	1467c	1505c	1508c	1531c	1540c	1554c	1574c	1596c	1600c	
	CTABLE(brstuff)		1623c	1781s	1806c												
	CTABLE(ctr)		934d	958d	1007c												
			325d	328d	330d	331d	332d	333d	334d	335d	336d	337d	338d	413c	473c	497c	
			668d	670c	673c	677d	679c	682c	686d	690c	694c	698d	702c	707c	711d	713c	
			718c	720d	724c	733c	737d	741c	752c	756d	764c	768c	772d	783c	787c	791d	
			795c	796c	797c	798c	805d	808c	809c	881c							
	CTABLE(options)		114d	118d	120d	123d	125d	127d									
	CTABLE(scanstuff)	***	1114c														
	DISCHPIB(dischpiib)		221c	223c	232c	239c	246c	261c	292c	315c	351c	363c	394c	426c	446c	508c	
			528c														
	ETU		637d	641c	698d	700c	701c	703c	704c								
	F9885(f9885dvr)	***	260c														
	INIT(intunits)	***	2208c														
	INITLOAD	***	1679c														
	INITLOAD(bootdammodule)		546d	778d	786c	787c	788c										
	INITLOAD(sysglobals)	***	146d														
	MUI		35c	199c	200c	350c	351c										
	PRINTER(prtdvr)	***	273c														
	SRAM(srammodule)		149c														
	SRMORAM(srmoramodule)		163c	417c													
	SRM_DRV(srm)		1230c	1253d	1256c	1297d	1304c	1315c	1328c	1330c	1336c	1338c	1378c	1393c	1415c	1417c	
			1420c	1422c	1435c	1440c	1445c	1450c	1460c								
			1009c	1017c	1092c												
	TAPEBKUP																
sc_list	CTABLE	***	1467c														
	CTABLE(options)	***	217d														
sc_list_length	CTABLE	***	1464c														
	CTABLE(options)		211d	215d													
sc_list_type	CTABLE(options)		214d	218d													
sc_table_entry_ptr	PRINTER(prtdvr)		67d	95c	111c	120c	208c	274c	288c								
scaler	GEN(vars(dgl_vars))	***	1293d														
	GEN(dgl_gen)		3290c	3440c													
scan_dav	CTABLE(scanstuff)		1040d	1083d	1114c	1115c	1116c	1117c									
	scanneddevice_letter																
	CTABLE		1347c	1364c	1471c												
	CTABLE(scanstuff)		1040d	1083d	1106c	1111c	1126c	1137c									
scanner_msus	CTABLE		1208d	1463c	1472c												
scanner_uap	CS80(cs80dsr)		614d	618c	619c	620c	621c	622c	626c	627c	628c						
scanptr	EPROMS(eproms)		39d	60c	61c	66c	68c	71c	80c	86c	88c	89c	95c	98c	100c	102c	
			106c	107c													
scanrec	EPROMS(eproms)		31d	39d													
scanstuff	CTABLE	***	1186d														
	CTABLE(scanstuff)	***	1025d														
scantitle	ETU	***	316c														
	INIT(fs)	***	757c														
	LIBRARIAN	***	2931c														
	M88KSYS(ci)	***	540c	918c													
	MUI	***	139c														
	TAPEBKUP	***	1069c														

scantitle_ok	MUI	129d	139c	140c													
	TAPEBKUP	1059d	1069c	1070c													
scat	SRM_DRV(srm)		878d	1556c													
scatpass	SRM_DRV(srm)		880d	1594c													
scg_base	DISCHPIB(dischpiib)		213d	282c	341c												
changeprotect	SRM_DRV(srm)		882d	1632c													
changevolume	SRM_DRV(srm)		884d	1663c													
sclear	A804XDVR(a804xdvr)		333d	350c	352c												
sclose	SRM_DRV(srm)		886d	1684c													
scode	EDRIVER(edriver)		36d	42d	45d	47d	74d	131c	132c	147c	306d	322c	325d	331c	334d	340c	
			343d	348c													
	ETU		69d	572c	578c	591c	593c	648c	660c	704c	720c	760c	763c	777c	778c	779c	
			780c	781c	783c	1071c	1231c										
scodes	ETU		696d	703c													
scopy	SRM_DRV(srm)		888d	1709c													
scratefile	SRM_DRV(srm)		890d	1743c													
scratelink	SRM_DRV(srm)	***	1793c														
	SRM_DRV(srm)	***	892d														
screen	CRT(crt)		164d	192c	272c	284c	321c	331c	332c	335c	341c	342c	344c	483c	585c	588c	
			605c	607c	608c	615c	619c	624c	628c	642c	644c	668c	675c	678c			
screenheight	CRT(crt)		160d	303c	670c												
	GCRT(crtb)		109d	212c	381c	386c	392c	395c	401c	403c	406c	504c	506c	507c			
screenwidth	CRT(crt)		163d	333c	343c	471c	473c	673c	675c								
	GCRT(crtb)		112d	342c	344c	507c											
screenwidth	CRT(crt)		159d	272c	300c	321c	331c	333c	335c	342c	343c	471c	472c	483c	576c	580c	
			582c	594c	603c	614c	619c	624c	628c	642c	644c	669c					
			108d	209c	342c	343c	503c	505c	507c								
scroll	GCRT(crtb)		338d	465c													
	GCRT(crtb)		245d	336c													
scrollup	CRT(crt)		328d	449c	459c	488c											
	GCRT(crtb)		240d	319c	339c	356c											
script	CRT(crt)		58d	158d	164d												
	GCRT(crtb)	***	94d														
scrttype	GCRT(crtb)		93d	94d													
scsize	INITLOAD(bootdammodule)		779d	786c	787c	788c											
scstype	INITLOAD(sysglobals)	***	199d														
sd	ETU		219d	240c	242c												
sdate	LIFDAM(lifdammodule)		56d	329c	331c	332c	333c	346c	347c	348c	349c	516c	573c				

II-147

[illegible]


```

serial_fastest
  IOLIB(general_4)      *** 1508c
  KERNEL(iodeclarations) *** 552d
  SRM_DRV(srm)          *** 1344c
serial_fhs
  DISCP1B(dischp1b)    *** 471c 510c
  IOLIB(general_4)      *** 1508c
  KERNEL(iodeclarations) *** 551d
serial_line
  IOLIB(serial_0)       2219d 2357d 2476c
serial_textam
  INIT(misc)            515d 697c
serial_textamhook
  INIT(misc)            470c 697c
  INITIORD(sysglobais) *** 279d
  M68KSYS(c1)           *** 334c
set_3v_address
  MINIT(cs80ir)         564d 818d 843c
  MINIT(qminit)         1179c 1209c 1228c 1240c
set_6845s
  GLE_KGL(gle_ras_out)  *** 8258c
set_acceleration
  DGL_HPGL(dgl_hpgli)  17111d 17169c
set_access_date
  SRM_DRV(srm)          *** 735d
set_address_and_return_mode
  TAPEBKUP(cs80tdvrr)   *** 670c
  TAPEBKUP(cs80tbr)     26d 100d 129c
set_all_color_table
  DGL_RAS(dgl_raster)   17247d 17537c
set_aspect
  LTB(dgl_lib)          20018d 20348d
set_auto_delay
  KEYS(keys)            *** 589c
  SYSDEVS(sysdevs)      *** 140d
set_auto_pen
  DGL_HPGL(dgl_hpgli)   17074d 17154c
set_auto_repeat
  KEYS(keys)            *** 588c
  SYSDEVS(sysdevs)      *** 140d
set_baud_rate
  IOLIB(serial_3)       2502d 2528d
set_char_length
  IOLIB(serial_3)       2508d 2681d
set_char_size
  LTB(dgl_lib)          20071d 20719c 20816d 21152c
set_color
  DGL_PLY(dgl_poly)     20215c 20245c 20599c 20840c
  LTB(dgl_lib)          20036d 20447d 21171c
set_color_model
  LTB(dgl_lib)          20034d 20399d
set_color_table
  LTB(dgl_lib)          20086d 20408d
set_creation_date
  SRM_DRV(srm)          *** 737d
set_display_lim
  LTB(dgl_lib)          20052d 20488d
set_echo_pos
  LTB(dgl_lib)          20074d 20846d
set_file_mask
  AMIGO(amigodvr)       *** 880c
  AMIGO(csamigo)        146d 274d
set_force
  DGL_HPGL(dgl_hpgli)   17103d 17166c

set_gator_vals
  DGL_COUT(dgl_conf_out) 11247d 11290c 11298c 11308c 11315c
set_hpb
  IOLIB(hpb_0)          1709d 1722d
  IOLIB(hpb_2)          1818c 1818c 1959c 1968c
set_in_use
  DISCP1B(bkgnd)        40d 89c 114c 119c 132c
set_kbdlang
  A804XDR(a804xdvr)     *** 397c
  SYSDEVS(sysdevs)      *** 142d
set_kbdtype
  A804XDR(a804xdvr)     *** 396c
  SYSDEVS(sysdevs)      *** 142d
set_line_style
  DGL_PLY(dgl_poly)     20216c 20246c 20948c 21019c
  LTB(dgl_lib)          20035d 20469d 21172c
set_line_width
  DGL_PLY(dgl_poly)     20217c 20247c 20949c 21021c
  LTB(dgl_lib)          20037d 20431d 21173c
set_locator_lim
  LTB(dgl_lib)          20083d 20880d
set_marker
  GLE_GEN(gle_gen)      *** 2154c
  GLE_HPGL(gle_hpgl_out) *** 7539c
  GLE_RSL(gle_ras_out)   *** 8407c
  GLE_TYPES(gle_types)   *** 1056d
set_options
  CS80(cs80)            346d 522d 534c
  CS80(cs80dsr)         *** 844c
  TAPEBKUP(cs80tdvrr)    *** 377c
set_parity
  IOLIB(serial_3)       2511d 2735d
set_pgn_color
  DGL_PLY(dgl_poly)     20014d 20102d
set_pgn_is
  DGL_PLY(dgl_poly)     20013d 20126d
set_pgn_style
  DGL_PLY(dgl_poly)     20020d 20142d 20187c
set_pgn_table
  DGL_PLY(dgl_poly)     20015d 20165d
set_polygon_color
  DGL_PLY(dgl_poly)     20053d 20850c
set_release
  CS80(cs80)            345d 500d 518c
  CS80(cs80dsr)         *** 882c
set_rate
  SYSDEVS(sysdevs)      178d 424c
set_rule
  GLE_RSL(gle_ras_out)  8120d 8228c 8229c 8239c 8282c
set_scannedevice_letter
  CTABLE(scanstuff)     1099d 1123c 1124c
set_serial
  IOLIB(serial_0)       2215d 2231d
set_specified_unitvol
  TAPEBKUP(cs80tdvrr)   *** 730c
  TAPEBKUP(cs80tbr)     24d 55d 69c
set_status_mask
  CS80(cs80)            347d 538d 552c
  CS80(cs80dsr)         888c 950c
set_stop_bits
  IOLIB(serial_3)       2505d 2597d
set_text_rot
  LTB(dgl_lib)          20070d 20720c 20785d 21157c

```


short																				
CTABLE(ctr)	***	403c																		
INITLOAD(loader)		926d	1272c	1442c	1526c															
LIBRARIAN		411c	1280c	1628c	1646c	1927c	1931c	1941c	2053c	2102c	2315c	2389c	2399c	2404c	2406c					
SEGMENTER(asm)	***	2412c	2414c	2501c	2581c	2582c	2621c	2651c	2715c	2762c	2780c	2810c								
short_defaults		135c																		
DGL_VARS(dgl_vars)	***	1320d																		
GEN(dgl_gen)		3284c	3456c	3458c																
short_flag																				
DGL_POLY(dgl_poly)	***	20975c																		
DGL_VARS(dgl_vars)	***	1317d																		
GEN(dgl_gen)		3411c	3462c																	
LIB(dgl_lib)		20624c	20655c																	
short_fc																				
DISCHPIB(dischpib)		209d	395c																	
short_vertex																				
DGL_POLY(dgl_poly)		20387d	20539c	20542c																

shortint																				
A804XDVR(a804xdrv)		65d	66d	126d	151d	152d	233d													
AMIGO(amigodvr)		428d	439d	443d	449d	492d	507d	558d	795d											
AMIGO(csamigo)	***	90d	123d	129d	130d	131d	244d													
CLOCK(clock)		47d																		
CRT(crt)		53d	163d	165d	167d	205d	206d	207d	298d	315d	316d	329d	339d	547d	550d					
CS80(cs80dvr)		551d	552d	553d	554d	713d	714d													
CTABLE		982d	985d	995d	1018d	1027d	1218d													
CTABLE(brstlff)	***	1212d	1216d	1268d	1292d	1374d														
CTABLE(ctr)		968d																		
CTABLE(options)		311d	313d	314d	325c	328d	329d	330d	331d	332d	333d	334d	335d	336d	337d					
CTABLE(scanstuff)		338d	339d	424d	425c	426d	427d	578d	580d	604d	615d	655d	668d	677d	686d					
C_HOOK		698d	711d	720d	737d	756d	772d	791d	805d	812d										
DGL_C_OUT(dgl_cfg_out)		136d	215d																	
DGL_RAS(dgl_raster)	***	1043d	1052d	1053d	1056d	1094d	1135d													
DISCHPIB(bknd)		49d	55d																	
DISCHPIB(dischpib)		11026d	11029d	11030d	11031d	11081d	11086d	11087d	11162d	11163d	11213d	11214d	11215d	11216d						
F9885(f9885dvr)	***	17384d																		
GCRT(crtb)		47d	79d	84d	95c	287d	357d	383d	388d	410d										
HPHIL(hphil)		196d	198d	199d	200c															
INIT(fs)	***	55d	94d	108d	112d															
INIT(ldr)		100d	108d	109d	110d	111d	112d	113d	114d	117d	121d	122d	126d	128d	129d					
INIT(misc)		130d	131d	132d	207d	224d	225d	226d	251d	372d	417d	419d	498d	529d	530d					
INITLOAD	***	531d																		
INITLOAD(bootoamodule)		228d																		
INITLOAD(loader)		748d	801d	803d	805d	806d	807d	809d	810d	811d	812d	814d	819d	820d	822d					
INITLOAD(mini)	***	827d	829d	831d	833d	835d	838d													
INITLOAD(sysglobals)		2386d	2408d																	
KEYS(keys)		181d	228d	247d	262d	265d	428d													
LIBRARIAN		1737d																		
M68KSYS(ci)		546d	560d	578d	589d	609d	649d	778d												
MINIT(asmr)		827d	828d	833d	835d	892d	1022d	1029d	1038d	1043d	1082d	1174d	1206d	1210d	1215d					
MINIT(hminit)		1220d	1225d	1400d	1401d	1423d	1589d													
MINIT(mminit)		319d	334d	373d	413d															
MINIT(xminit)		30d	77d	96d	104d	124d	154d	170d	239d	294s										
MUI		46d	25d	37d	151d	441d	448d	478d	755d	1273d	1476d	1486d	1655d	1656d	1718d					
MOREFSYS(mfs)		1719d	1720d	1746c	1852d	1999d	2000d	2001d	2032c	2309d	2669c	2722d	2734d	2829d	2832d					
PRINTER(prtdvr)		2834d	2903d	3015d	3016d	3027d	3055d	3247d	3270d	3289d	3306d									
SRM_DRV(srm)		112d	313d	872d																
SRM_DRV(srm)		1331d	1335d	1337d	1342d	1346d	1348d													
SRM_DRV(srm)		254d	299d	304d	345d	346d	347d	350d	351d	352d	412d	413d	414d							
SRM_DRV(srm)		116d	143d	150d																
SRM_DRV(srm)		885d	1038d																	
SRM_DRV(srm)		1365d	1366d	1384d																
SRM_DRV(srm)		223d	286d	395d	398d	399d	401d	402d	410d	458d										
SRM_DRV(srm)		19d	22d	35d	38d	57d	401d	536d	553d	560d	564d	667d								
SRM_DRV(srm)		77d	87d																	
SRM_DRV(srm)		76d	100d	912d	961d	962d	1020d	1021d												
SRM_DRV(srm)		215d	216d	997d	1405d															
SRM_DRV(srm)		49d	59d	60d	88d	91d	128d	129d	133d	134d	135d	320d	321d	322d	323d					
SRM_DRV(srm)		598d																		
SRM_DRV(srm)		1110d	1164d	1225d																
SRM_DRV(srm)	***	45d																		
SRM_DRV(srm)	***	55d																		
shortpatch																				
LIBRARIAN		2368d	2447c	2457c																
shortpointer																				
HEAPT(hpm)		93d	96c	140c	141c	270c	311c													
showprompt																				
ETU	***	111d																		
shuffle																				
LIFDAM(lifmod.le)		683d	907c																	

[illegible]

size_from_set_date		
SRM_DRV(srm)	***	109d
size_from_set_eof		
SRM_DRV(srm)	***	108d
size_from_volstatus		
SRM_DRV(srm)	***	110d
size_from_write		
SRM_DRV(srm)	***	111d
size_from_xchg_data		
SRM_DRV(srm)	***	112d
size_from_xchg_open		
SRM_DRV(srm)	***	113d
size_to_are_you_alive		
SRM_DRV(srm)		57d 1534c
size_to_cat		
SRM_DRV(srm)		58d 1561c
size_to_catprotect		
SRM_DRV(srm)		59d 1599c
size_to_change_vol_label		
SRM_DRV(srm)		62d 1666c
size_to_changeprotect		
SRM_DRV(srm)		60s 1638c
size_to_close		
SRM_DRV(srm)		63d 1687c
size_to_copy		
SRM_DRV(srm)		64d 1712c
size_to_create		
SRM_DRV(srm)		65s 1750c
size_to_createlink		
SRM_DRV(srm)	***	1799c
SRM_DRV(srm)	***	67d
size_to_flock		
SRM_DRV(srm)	***	1889c
SRM_DRV(srm)	***	68d
size_to_funlock		
SRM_DRV(srm)	***	2051c
SRM_DRV(srm)	***	69d
size_to_gang_cleanup		
SRM_DRV(srm)	***	1870c
SRM_DRV(srm)	***	70d
size_to_info		
SRM_DRV(srm)	***	1851c
SRM_DRV(srm)	***	71d
size_to_init_vol		
SRM_DRV(srm)	***	72d
size_to_open		
SRM_DRV(srm)	***	1916c
SRM_DRV(srm)	***	73d
size_to_position		
SRM_DRV(srm)	***	1949c
SRM_DRV(srm)	***	74d
size_to_purge		
SRM_DRV(srm)	***	1977c
SRM_DRV(srm)	***	75d
size_to_read		
SRM_DRV(srm)	***	2005c
SRM_DRV(srm)	***	76d
size_to_set_date		
SRM_DRV(srm)	***	78d
size_to_set_eof		
SRM_DRV(srm)	***	2030c
SRM_DRV(srm)	***	77d

size_to_volstatus		
SRM_DRV(srm)	***	2069c
SRM_DRV(srm)	***	79d
size_to_write		
SRM_DRV(srm)	***	2091c
SRM_DRV(srm)	***	80d
size_to_xchg_data		
SRM_DRV(srm)	***	81d
size_to_xchg_open		
SRM_DRV(srm)	***	1832c
SRM_DRV(srm)	***	82d
sizechk		
LIFDAM(lifmodule)		99d 102c
skip		
SRMDAM(srmdammodule)	1528d	1550c 1555c 1573c 1576c
skipfor		
TOLIB(general_2)	821d	1070d
skipping		
M68KSYS(c1)	982d	1061c 1074c 1076c 1081c 1086c 1145c
sl		
ETU		176d 179c 180c 182c
GLE_UTILS(gle_utils)	21010d	21106d 21113c 21114c
LIFDAM(lifmodule)	134d	139c 140c 144c
SEGENER (asm)	54d	131c 266c
slast		
UCSD_DAM(ucsdmodule)	185d	200c 205c 239c
slink		
CTABLE(ctr)	448d	454d 461c 464c
CTABLE(scanstuff)	***	1061d
slist		
DGL_INQ(dgl_inq)	6012d	6088d 6441c 6450c 6461c 6475c 6488c 6503c 6514c 6530c 6543c
LIBRARIAN	1716d	1729c 1754c 1771c 1781c 1786c 1997d 2015c 2039c 2059c 2070c 2080c 2121c
slock		
SRM_DRV(srm)	***	1886c
SRM_DRV(srm)	***	902d
slop		
INITLOAD(loader)	1138d	1147c 1148c 1156c 1158c
slowterm		
CRT(crt)	***	113d
GCRT(crtb)	***	46d
SYSDEVS(sysdevs)	***	69d
slp		
INITLOAD(loader)	***	908d
LIBRARIAN		1729c 2015c 2121c
small		
KEYS(keys)	376d	442c 489c 491c 510c 512c
smallest		
HEAPT(hpm)	55d	59d 60d 122d 123d 143c 144c 230c 293c 310c
smallkbd		
A804XDVR(a804xdvr)		216c 223c
KEYS(keys)	***	442c
NONUSKBD1(non_us_kbd1)	***	118c
SYSDEVS(sysdevs)	***	145d
smallsize		
HEAPT(hpm)	70d	104c 114c 116c 122d 143c
smh		
SRM_DRV(srm)	1477d	1482c
sn		
CTABLE(ctr)	339d	812d 815c 816c
so		
CS80(cs80)	524d	530c 531c 532c
soft_clip_cpx		
GLE_TYPES(gle_types)	***	1189d

[illegible]

[illegible]

II-159

status_bytes	444d	476c	478c	482c	484c	509d	527c	529c	532c	534c	656d	661c	662c	792d
AMI0(amigodvr)	801c	803c												
AMI0(csamigo)	146d	304d	312c	313c	317c	321c								
CS80(cs80)	342d	457d	466c	467c										
CS80(cs80dsr)	640d	650c	658c	682c	755c	757c	763c							
MINIT(hminit)	302d	316c	317c	325c	470c	478c								
TAPBACKUP(cs80tbdvr)	520d	530c	538c	562c	637c	639c	645c	647c	653c	700d	758c	766c	794c	871c
	874c	878c	880c	886c										
status_cmd_buf														
AMI0(csamigo)	306d	310c												
status_def														
GLE_TYPES(gle_types)	1243d	1273d												
status_mask														
CS80(cs80)	347d	538d	549c											
status_mask_type														
CS80(cs80)	201d	211d	347d	538d	544d									
CS80(cs80dsr)	***	795d												
status_type														
AMI0(amigodvr)	444d	509d	656d	792d										
AMI0(csamigo)	95d	148d	157d	304d										
CS80(cs80)	205d	342d	362d	457d										
CS80(cs80dsr)	***	640d												
MINIT(hminit)	***	302d												
TAPBACKUP(cs80tbdvr)	***	520d	700d											
status_line														
SYSDVS(sysdevs)	248d	582c	585c	644c										
status_reg														
A804XDVR(a804xdvr)	62d	94c	119c	134c	383c	386c								
status_type														
A804XDVR(a804xdvr)	51d	114d	132d											
BAT(bat)	39d	52d	60d											
status_hl														
HEAPT(hpm)	56d	57d	59d	231c										
status_hl														
HEAPT(hpm)	***	59d												
status														
LIFDAM(lifmodule)	163d	183c	184c	189c										
steps_size														
EDRIVER(edriver)	83d	171c	178c	182c	186c									
EPROMS(eproms)	17d	95c	98c											
status														
F9885(f9885dvr)	***	88c												
status														
F9885(f9885dvr)	***	88c												
status														
SYSDVS(sysdevs)	***	451d												
status														
CTABLE(scanstuff)	***	1088d												
status														
CRT(crt)	99d	145d												
GCR1(crtb)	***	78d												
SYSDVS(sysdevs)	***	94d												
stopaction														
KEYS(keys)	59d	516c												
stopgoing														
M68KSYS(ci)	870d	891c	902c	904c	909c									
stopline														
MIUI	224d	260c	261c	270c										
TAPBACKUP	1111d	1147c	1148c	1157c										
storage														
CTABLE(scanstuff)	1087d	1121c												
store_dit														
GLE_RGL(gle_res_out)	8208c	8215c												

stop														
INITLOAD(loader)	902d	1187c	1265c	1289c	1290c	1291c	1309c	1348c						
LIBKARIAN	***	481c												
str9														
INITLOAD(loader)	1585d	1587d												
str_count	1242d	1614d	1617c	1622c	1623c	1629c	1633c							
LOL18(general_4)														
strcnt	6166d	6456c	6457c	6458c	6460c	6462c	6483c	6484c	6485c	6487c	6489c			
DGL_INQ(dgl_inq)														
stream	1031d	1071c	1135c											
M68KSYS(ci)														
streamchain	58d	96c	1035c	1071c										
M68KSYS(ci)														
streamdvr	306d	547c	548c											
M68KSYS(ci)														
streamfib	54d	89c	321c	322c	330c	331c	342c	348c	349c	357c	367c	368c	372c	472c
M68KSYS(ci)	484c	499c	510c	514c	518c	545c	546c	550c	862c	863c	945c	1054c	1055c	
streaming														
ETU	106c	126c	133c	197c	206c	228c	552c	814c	1007c	1257c				
LIBKARIAN	90c	273c	2093c	3406c	3683c									
M68KSYS(ci)	74d	88d	90c	1076c	1081c	1142c								
streamopen														
M68KSYS(ci)	387d	1056c	1180c	1181c										
streamsyntax														
M68KSYS(ci)	403d	435c	530c	543c										
strect														
MOREFSYS(mfs)	56d	93c	94c	95c	377c									
stretch														
UCSD_DAM(ucsdmodule)	352d	642c												
stretchf														
LIFDAM(lifmodule)	1014d	1132c												
stretchhit														
ASCII(asciimodule)	***	81c												
INIT(misc)	***	433c	498c	617c										
INITLOAD(sysglobals)	***	136d												
LIFDAM(lifmodule)	***	1132c												
SRMDAM(srmdammodule)	***	1666c	1787c											
UCSD_AM(ucsd_am)	***	57c												
UCSD_DAM(ucsdmodule)	***	642c												
str9														
MOREFSYS(mfs)	30d	89c	544c											
strin														
KEYS(keys)	65d	66d												
string2														
INITLOAD	***	1758d												
INITLOAD(bootdammodule)	538d	546d	547d	778d	782d	795d								
string255														
CRT(crt)	***	505d												
CTABLE(ctr)	***	316d	382d	422d	423d	506d								
ETU	***	422d												
GCR1(crtb)	***	373d												
INITLOAD(bootdammodule)	***	764d	766d	817d	820d	825d	827d	830d	832d	836d				
INITLOAD(sysglobals)	***	552d	588d	648d										
LIBKARIAN	***	35d	37d											
LIFDAM(lifmodule)	***	3302d												
M68KSYS(ci)	***	98d	109d	1060d										
MINIT(midecs)	***	308d	391d	455c										
MINIT(qminit)	***	37d	41d											
MIUI	***	881d	932d	936d	1034d	1036d								
MOREFSYS(mfs)	***	239d	124d	131d	290d									
SRMDAM(srmdammodule)	***	21d	46d	533d	554d	563d	669d							
UCSD_DAM(ucsdmodule)	***	98d	110d											
	***	42d												

II-160

[illegible]

s type														
INIT(fs)	***	766d												
MOREFSYS(mfs)		44d	64c	75c	89c									
sub														
LIBRARIAN		156d	158c	1482d	1486d	1492c	1495c	1505c	1523c	1547c	1565c	1566c	1574c	1579c
sub_poly_count														
DGL_POLY(dgl_poly)		20925d	20955c	20968c										
sub_poly_start														
DGL_POLY(dgl_poly)		20924d	20953c	20964c	20965c	20982c								
subit														
INITLOAD(loader)	***	865d												
LIBRARIAN		191c	193c	1505c	1523c	1574c	1579c	2395c						
subset80														
CS80(cs80)	***	81d												
MINIT(qminit)		973c	1063c	1074c	1162c									
success														
SRRDAM(srrdammodule)	***	1479c												
SRM_DRV(srm)	***	836d												
suffix														
INIT(fs)	***	756d												
LIFDAM(lifmodule)		142c	168c											
suffixtable														
ASCII	***	320c												
INIT(misc)		701c	705c	714c	722c	726c								
INIT_LOAD(sysglobals)	***	263d												
LIFDAM(lifmodule)		147c	148c	152c	176c	180c	187c	188c						
UCSD_AM(ucsd_am)	***	270c												
suffixtype														
INIT_LOAD(sysglobals)		36d	169d											
sufstableptrtype			174d	263d										
INIT_LOAD(sysglobals)														
sufstabletype			169d	174d										
INIT_LOAD(sysglobals)														
sunlock														
SRM_DRV	***	2048c												
SRM_DRV(srm)	***	916d												
sup_dev_type														
HUII		15d	20d	318d										
suppress_release_timeout														
CS80(cs80)	***	231d												
surface_ert														
MINIT(cs80ir)	***	551d												
surface_mode														
AMIGO(amigodvr)	***	737c												
AMIGO(csamigo)		139d	183d	185c	291c	349c	371c							
suvs														
TAPEBKUP(cs80tbr)		63d	65c	66c	67c									
svv_cmd_pair														
MINIT(cs80ir)		583d	585c	586c	612c	666c	692c	718c	750c	790c	835c	863c		
TAPEBKUP(cs80tbr)		48d	50c	51c	90c	120c	148c	171c	193c	274c				
sva														
CS80(cs80)		572d	583c	584c										
TAPEBKUP(cs80tbr)		117d	125c	126c										
sva_type														
CS80(cs80)		86d	115d	215d	217d	221d	222d	224d	225d	572d				
TAPEBKUP(cs80tbdvr)	***	338d												
TAPEBKUP(cs80tbr)		28d	117d	226d	231d	255d								
svdate														
LIFDAM(lifmodule)		324d	327c	338d	344c									
svol														
SRM_DRV	***	2066c												
SRM_DRV(srm)	***	918d												
svol_bytes														
CTABLE(ctr)		311d	595d	600c	658c	693c	694c	706c	707c	716c	729c	746c		

II-161

II-162


```

sysdevs
A804XDVR(a804xdvr)      ***      31d
BAT                      ***      99d
CLDCK(clock)            ***      33d
CRT(crt)                ***      33d
C_HOOK                  ***      35d
DGL_C_OUT(dgl_confg_out) *** 11004d
DGL_RAS(dgl_raster)     *** 17012d
ETU                      ***      26d
GCRt(crtb)              ***      33d
GLE_RGL(gle_ras_out)    ***      8004d
HPHIL(hphil)            ***      30d
KEYS(keys)              ***      33d
LIB(dgl_lib)            *** 20097d
LIBRARIAN                ***      27d
LIFDAM(lifmodule)       ***      26d
M68KSYS(c1)              ***      33d
MINIT(mminit)           ***     111d
MIUI                     ***      11d
MOREFSYS(mfs)           ***      10d
MOUSE(mouse)            ***     29d
NONUSKBD1(non_us_kbd1)  ***      31d
NONUSKBD2                ***      30d
PRINTER(prtdvr)         ***      37d
SYSDEVS                  ***     661d
SYSDEVS(sysdevs)        ***      29d
TAIL                     ***      32d
TAPEBKUP                 ***     953d
UCSD_AM(ucsd_am)         ***      29d
UCSD_DAM(ucsdmodule)    ***      29d
sysescapcode
INIT(misc)               ***     234c   241c
INITLDRD(sysglobals)    ***     239d   866c
M68KSYS(c1)              ***     859c   1170c
TAPEBKUP                 ***
sysfile
INIT(misc)               ***     726c   727c
INITLDRD(bootdammodule) ***     671c
INITLDRD(sysglobals)    ***     52d
LIBRARIAN                ***     3140c
sysfilenames
M68KSYS(c1)              41d      55d   581d
sysfiles
M68KSYS(c1)              39d      40d   41d   586d   587d   591d   657d   661d
sysfilevols
M68KSYS(c1)              40d      583d
sysflag
CRT(crt)                 ***     238c   253c   687c   720c
DGL_C_IN(dgl_confg_in)  *** 12050c
DGL_TOOLS(dgl_tools)    *** 20028d 20032c 20034c 20061c
INITLDRD(sysglobals)    ***     283d
KERNEL(general_0)       ***     1015c
sysflag2
INITLDRD(sysglobals)    ***     289d
sysflag_def
C_HOOK                   38d      52d
DGL_TOOLS(dgl_tools)    20016d 20028d

```

```

sysglobals
A804XDVR(a804xdvr)      ***      31d
AMIGO(amigodvr)         ***     425d
AMIGO(csamigo)           ***      37d
ASCII                    ***     317d
ASCII(asciimodule)      ***      4d
BAT(bat)                 ***      33d
BUBBLES(bubble)         ***      26d
CLOCK(clock)             ***      33d
CONVERT(convert_text)   ***      27d
CRT(crt)                 ***      33d
CS8D(cs80)               ***     66d
CS8D(cs80dsr)            ***     598d
CS8D(cs80dvr)            ***     972d
CS8D(tapebuf)            ***      35d
CTABLE                    ***     1186d
CTABLE(brstuff)          ***     928d
CTABLE(ctr)              ***     239d
CTABLE(options)          ***      47d
CTABLE(scanstuff)        ***    1035d
C_HOOK                   ***      35d
DC_DRV(extdc)            ***     194d
DC_DRV(intdc)            ***     594d
DC_DRV(intdc)            ***     232d
DGL_C_IN(dgl_confg_in)   ***    12017d
DGL_C_OUT(dgl_confg_out) ***    11017d
DGL_RAS(dgl_raster)      ***    17014d
DISCHPIB(bkgnd)          ***      31d
DISCHPIB(dischpib)       ***     189d
DI_DRV(extdi)             ***     150d
DI_DRV(intdiscint)        ***     200d
DMA_DRV(int_dma)          ***     164d
EDRIVER(edriver)         ***      24d
EPRDMS(eproms)           ***      6d
ETU                       ***     26d
F9885(f9885dvr)         ***     37d
GCRt(crtb)               ***     33d
GLE_KNOB(gle_knob_in)    ***    18028d
GLE_RGL(gle_ras_out)     ***     8004d
G_DRV(extg)               ***     167d
G_DRV(int_gpio)           ***     218d
HEAPT(sysglobals)        ***      26d
HPHIL(hphil)             ***      30d
H_DRV(exth)               ***     166d
H_DRV(int_hpib)           ***     2514d
INIT                      ***     739d
INIT(fs)                  ***    2135d
INIT(intunits)           ***      34d
INIT(isr)                 ***    2270d
INIT(idr)                 ***     153d
INIT(misc)                ***     164d
INITLDRD                  ***     535d
INITLDRD(bootdammodule)  ***     809d
INITLDRD(loader)         ***     304d
INITLDRD(mini)            ***      6d
INITLDRD(sysglobals)     ***     826d
IDLIB(general_2)         ***     771d
KERNEL(general_0)        ***     236c
KERNEL(lodeclarations)   ***      33d
KEYS(keys)                ***    20096d
LIB(dgl_lib)              ***      27d
LIBRARIAN                 ***     26d
LIFDAM(lifmodule)        ***     27c
LOCKMOD(lockmodule)      ***     22c
M68KSYS(c1)              ***

```


II-164

t	CLOCK(clock)	74d	76c	79c	80c	81c	137d	142c										
	CONVERT(convert_text)	35d	36d	40d	65c	72c	76c	79c	87d	100c	104c	117c	119c					
	CS80(cs80)	345d	500d	513c														
	CTABLE(ctr)	345d	639c	645c	651c	658c	664c	673c	682c	694c	707c	716c	733c	752c	768c			
	DGL_POLY(dgl_poly)	787c	809c	816c														
	GLE_RGL(gle_ras_out)	20411d	20795c	20797c	20798c	20834c	20836c	21032d	21040c	21078d	21086c	21123d	21130c	21154d	21161c			
	GLE_UTILS(gle_utils)	8160d	8163d	8164d	8165d	8166d	8167d	8168d	8169d	8170d	8171d	8172d	8173d	8174d	8175d			
	HEAPT(hpm)	8176d	8177d	8178d														
	INIT(fs)	21044d	21035c	21061c	21066c													
	M68KSYS5(ci)	87d	89c	90c	94d	96c												
	MOREFSYS(mfs)	795d	796d	800d	801d	802d	803d	804d	805d	806d	807d	808d	809d	810d	811d			
	SYSDEVS(sysdevs)	812d	813d	814d	825d	827d												
		398d	540c															
		18d	19d	515d	520c	523c	526c	667d	675c									
		474d	475c															
t1	CLOCK(clock)	90d	94c	96c														
	NONUSKBD1(non_us_kbd1)	76d	77d	82d	85d	88d	91d	94d										
	NONUSKBD2	33d	34d	61d	63d	66d	70d	74d	78d	82d	87d	93d	99d	103d	108d			
		110d	114d	117d	119d													
t2	CLOCK(clock)	90d	95c	96c														
	NONUSKBD1(non_us_kbd1)	77d	80d															
	NONUSKBD2	34d	60d															
t3	NONUSKBD2	35d	36d	40d	41d	42d	43d	44d	45d	46d	47d	48d	49d	50d	54d			
		55d	56d	57d	58d													
t4	NONUSKBD2	36d	39d															
t_cnt	IOLIB(general_4)	1203d	1355d	1371c	1396c	1410c	1426c											
t_count	IOLIB(general_4)	1229d	1258d	1267c	1271c	1442d	1444c	1445c	1466d	1468c	1469c	1491d	1493c	1494c	1534d			
		1538c	1541c	1543c														
t_dir	DISCHPIB(dischpib)	443d	453c	458c	474c													
	IOLIB(general_4)	1201d	1207d	1213d	1220d	1225d	1353d	1389c	1424c	1438d	1445c	1462d	1469c	1488d	1494c			
		1496c	1531d	1536c	1543c													
t_tfr	IOLIB(general_4)	1200d	1206d	1212d	1219d	1224d	1352d	1421c	1437d	1445c	1461d	1469c	1487d	1494c	1508c			
		1509c	1510c	1511c	1530d	1543c												
t_type	CS80(cs80)	229d	345d	500d	506d													
ta	CS80(cs80)	***	221d															
tab	INIT(misc)	***	164d															
	KEYS(keys)	***	481c															
table	TAIL		34d	50c	62c	75c												
tablename	TAPEBKUP(cs80tbr)		269d	277c														
tablepc	LIBRARIAN		309d	1154c	1220c	1233c												
tableptr	INITLOAD(loader)		1218d	1236c	1238c	1243c												
tag_base	DISCHPIB(dischpib)		212d	278c	338c	402c												
tagbase	PRINTER(prtdvr)		58d	220c														
tail	TAIL	***	30d															

talk	IOLIB(hpib_2)		1797d	1981d														
talk_constant	GLE_HPIB(gle_hpib_io)		10213c	10301c														
	IOLIB(hpib_1)		535c	623c														
	IOLIB(hpib_2)		1827c	1905c	1984c													
	KERNEL(io declarations)	***	316d															
talker	IOLIB(hpib_3)		2041d	2145d	2148c													
tape_dt	CTABLE		1377d	1381c														
tape_up	TAPEBKUP		1218d	1386c	1387c	1393c	1397c	1405c										
tapekup	TAPEBKUP	***	8d															
tapebuf	CS80	***	1411d															
	CS80(cs80dsr)	***	598d															
	CS80(cs80dvr)	***	972d															
	CS80(tapebuf)	***	32d															
	TAPEBKUP	***	953d															
	TAPEBKUP(cs80tbdvr)	***	305d															
tapebuf_block	CS80(cs80dvr)		1157c	1172c	1177c	1185c	1193c											
	CS80(tapebuf)	***	47d															
tapebuf_maxsize	CS80(cs80dvr)	***	1241c		41d													
	CS80(tapebuf)		39d															
tapebuf_ptr	CS80(cs80dvr)		1157c	1193c	1268c	1283c	1295c	1318c										
	CS80(tapebuf)		44d	55c	56c													
tapebuf_size	CS80(cs80dvr)		1157c	1192c	1193c													
	CS80(tapebuf)	***	48d															
tapebuf_state	CS80(cs80dsr)	***	629c															
	CS80(cs80dvr)		1149c	1156c	1158c	1172c	1178c	1186c	1194c	1294c	1306c	1317c	1369c					
	CS80(tapebuf)		45d	57c														
	TAPEBKUP		1245c	1482c														
	TAPEBKUP(cs80tbdvr)		583c	815c														
tapebuf_state_type	CS80(tapebuf)		42d	45d														
tapebuf_type	CS80(tapebuf)		41d	44d														
tapebuf_up	CS80(cs80dsr)	***	628c															
	CS80(cs80dvr)		1150c	1157c	1172c	1176c	1185c	1193c	1369c	1377c								
	CS80(tapebuf)	***	46d															
	TAPEBKUP(cs80tbdvr)		582c	814c														
tapesetup	TAPEBKUP(cs80tbdvr)		366d	383c														
target_h	DGL_RAS(dgl_raster)		17071d	17105c	17116c	17117c												
target_l	DGL_RAS(dgl_raster)		17073d	17105c	17121c	17125c												
target_s	DGL_RAS(dgl_raster)		17072d	17105c	17120c	17137c												
tb	GEN(dgl_gen)		3193d	3229c	3233c	3240c												
tbit	CS80(cs80)		506d	513c														
tc	DISCHPIB(dischpib)		270d	277c	32nd	336c												

tct														
GLE_HPGL(gle_hpgl_out)	7166d	7181c	7183c	7184c	7185c	7186c	7187c	7188c	7222d	7241c	7243c	7244c	7245c	7246c
GLE_HPGLI(gle_hpgl_in)	7247c	7248c												
tct_message	18097d	18107c	18109c	18110c	18111c	18112c	18113c	18114c	18134d	18147c	18149c	18150c	18151c	18152c
***	18153c	18154c	18291d	18300c	18302c	18303c	18304c	18305c						
td	1910c													
***	304d													
AB04XDVR(a804xdvr)	256d	267c	269c	274c	276c	279c	280c	296c	298c	302c	307c	308c	309c	316c
LIBRARIAN	317c	319c	320c											
***	2196d													
SYSDEVS(sysdevs)	221d	477d	478c	480c										
tdata														
AB04XDVR(a804xdvr)	257d	269c	276c	279c	282c	283c	289c	293c	296c	300c	302c	305c	307c	308c
***	309c													
tdate														
INITLOR0(bootdamodule)	556d	563d												
LIFDAM(lifmodule)	41d	54d	56d	64d	113d	548d								
tdp														
INITLOAD(loader)	900d	1528c												
LIBRARIAN	1282c	1870c	1873c	1877c	1883c	1913c	1921c	1943c	1947c	2171c	2199c	2624c	2628c	2765c
tdx														
DGL_POLY(dgl_poly)	20446d	20452c	20457c	20467c	20468c									
tdy														
DGL_POLY(dgl_poly)	20446d	20452c	20457c	20467c	20468c									
te														
INITLOAD(mini)	332d	338d	342d	343d	344d	345d	346d	347d	348d	349d	350d	351d	352d	353d
***	354d	355d	356d	357d	358d	359d	360d	361d	362d	363d	364d	365d	366d	367d
***	368d	369d	370d											
tea														
CTABLE(ctr)	420d	633c	639c	645c	651c	658c	664c	673c	682c	694c	707c	716c	733c	752c
***	768c	787c	809c	816c										
tea_amigo_mv														
CTABLE	1554c	1693s	1700s	1711s	1716s									
CTABLE(ctr)	334d	737d												
tea_amigo_sv														
CTABLE	1540c	1661s	1671s	1681s	1727s	1738s								
CTABLE(ctr)	335d	720d												
tea_boot														
CTABLE(ctr)	324d	662d												
tea_bubble														
CTABLE	***	1781s												
CTABLE(ctr)	338d	805d												
tea_crt														
CTABLE	***	1499c												
CTABLE(ctr)	326d	643d												
tea_csR0_mv														
CTABLE	1574c	1754s												
CTABLE(ctr)	336d	756d												
tea_csR0_sv														
CTABLE	1596c	1600c	1764s	1769s	1774s									
CTABLE(ctr)	337d	772d	796c											
tea_eprom														
CTABLE	***	1787s												
CTABLE(ctr)	339d	812d												
tea_fipv														
CTABLE	1274c	1278c												
CTABLE(ctr)	333d	791d												
tea_hpR290x														
CTABLE	1643s	1644s												
CTABLE(ctr)	332d	711d	795c											
tea_hpR885														
CTABLE(ctr)	330d	686d	797c											

tea_hpR895														
CTABLE	1531c	1648s	1649s	1654s										
CTABLE(ctr)	331d	698d	798c											
tea_kbc														
CTABLE	***	1500c												
CTABLE(ctr)	327d	649d												
tea_local_printer														
CTABLE	***	1508c												
CTABLE(ctr)	328d	677d												
tea_memory_volume_dam														
CTABLE	***	1497c												
CTABLE(ctr)	323d	637d												
tea_minir														
CTABLE	1638s	1639s												
CTABLE(ctr)	329d	655d	794c											
tea_nounit														
CTABLE(ctr)	631d	826c	870c											
tea_srm														
CTABLE	1505c	1623c												
CTABLE(ctr)	325d	668d												
temp														
AB04XDVR(a804xdvr)	379d	383c	396c	397c										
CRT(crt)	554d	642c	645c											
DC_DRV(extdc)	198d	199d	200d	202d	204d	205d	207d	209d	211d	212d				
DC_DRV(intdc)	244d	245d	246d	247d	248d	249d	250d	252d	254d	263d	267c	269c	273c	305d
	307c	312d	317c	322d	325c	330d	335c	340d	343c	348d	351c	356d	374c	382c
	389c	390c	398c	408d	417c	445c	450c	464c	494c	551c	556c			
DGL_POLY(dgl_poly)	20258d	20259c	20260c	20261c	20262c	20263c								
DI_DRV(extdi)	155d	156d	157d	158d	159d	160d	161d	163d	165d	166d	167d	168d	169d	170d
172d														
ETU	72d	730c	732c	733c	948c	952c	960c	961c	966c	1231c				
GEN(dgl_gen)	3390d	3493c	3494c	3495c	3517c	3518c	3519c	3552c	3554c	3557c	3567c	3569c	3573c	
GLE_HPGL(gle_hpgl_out)	7167d	7192c	7194c	7198c	7200c									
GLE_HPGLI(gle_hpgl_in)	18098d	18118c	18120c	18124c	18126c									
GLE_RSL(gle_ras_out)	8321d	8359c	8361c	8362c	8364c									
DC_DRV(extdc)	172d	173d	174d	175d	176d	177d	178d	180d	182d	183d	184d	185d		
HEAPT(hpm)	218d	223c	224c	226c	233c	234c	236c	237c	248c	257c	259c			
HPHIL(hphil)	227d	252c	253c	256c	257c	259c	260c	262c	264c	265c				
H_DRV(exth)	170d	171d	172d	173d	174d	175d	176d	178d	180d	181d	182d	183d	184d	185d
187d														
KERNEC(general_0)	803d	809d	811c	814d	816c	819d	821c	824d	826c	829d	831c	834d	836c	839d
841c		844d	846c											
KERNEC(iocomasm)	701d	702d												
KERNEC(iodeclarations)	418d	419d												
LIBRARIAN	1181d	123c	1234c	423d	425d	427d	430d	433d	435d	438d	440d			
LIFDAM(lifmodule)	738d	765c	767c	768c	769c	775c	787c	789c	790c	791c	797c	992d	997c	999c
1006c		1007c												
RS_DRV(rs)	172d	173d	174d	175d	176d	177d	178d	180d						
UCSD_IAR(ucsdmodule)	187d	243c	251c	256c	257c	381d	389c	390c	182d					
temp1														
SYSDEVS(sysdevs)	514d	549c	552c	553c										
temp2														
LIFDAM(lifmodule)	163d	186c	187c	188c	189c									
temp_color_map														
GLE_RSL(gle_ras_out)	8319d	8336c	8346c											
temp_color_model														
DGL_RHS(dgl_raster)	17582d	17619c	17645c											
temp_color_raster														
DGL_HPGL(dgl_hpgl)	17257d	17289c	17290c											
DGL_RHS(dgl_raster)	17580d	17608c	17609c											
temp_dav														
CTABLE	1341d	1344c	1345c	1347c	1348c	1349c								
temp_file_pass														
SAMDAM(srmdammodule)	65d	160c	1067c	1094c	1142c	1158c	1163c	1301c	1397c	1400c				

II-166

[illegible]


```

twsosets
MOUSE (mouse)          ***      42d
SYSDEVS(sysdevs)      ***      314d

tx
  GEN(dgl_gen)          3280d  3292c  3293c
  LIB(dgl_lib)          20267d 20277c 20281c

txmax
  LIB(dgl_lib)          20495d 20512c 20519c 20522c 20888d 20908c 20915c 20920c

txmin
  LIB(dgl_lib)          20494d 20511c 20519c 20522c 20887d 20907c 20914c 20920c

ty
  GEN(dgl_gen)          3280d  3294c  3295c
  LIB(dgl_lib)          20267d 20278c 20282c

tymax
  LIB(dgl_lib)          20497d 20515c 20520c 20522c 20890d 20910c 20917c 20920c

tymin
  LIB(dgl_lib)          20496d 20514c 20520c 20522c 20889d 20909c 20916c 20920c

typ
  INIT(fs)              764d   765d

type_card_id
  KERNEL(iodeclarations) 382d   543d

type_device
  GLE_HPIB(gle_hpib_io) 10188d 10278d
  IOLIB(general_2)      797d   799d   802d   806d   808d   811d   814d   817d   820d   822d   832d   913d   936d   966d
                        992d   1012d  1025d  1038d  1051d  1071d
  IOLIB(general_4)      1199d  1205d  1211d  1218d  1223d  1351d  1436d  1460d  1486d  1529d
  IOLIB(hpib_1)         403d   406d  408d  411d  510d  598d  696d  744d
  IOLIB(hpib_2)         1780d  1783d  1787d  1789d  1792d  1793d  1799d  1838d  1861d  1897d  1916d  1935d  1951d  1989d
  IOLIB(hpib_3)         2033d  2094d
  KERNEL(iodeclarations) ***      369d

type_hpib_addr
  GLE_HPIB(gle_hpib_io) ***      10120d
  IOLIB(hpib_1)         396d   444d
  IOLIB(hpib_2)         1782d  1796d  1798d  1854d  1974d  1982d
  KERNEL(iodeclarations) ***      387d

type_hpib_line
  IOLIB(hpib_0)         1710d  1712d  1714d  1723d  1734d  1745d
  KERNEL(iodeclarations) ***      388d

type_isc
  DC_DRV(init_dc)       ***      602d
  DC_DRV(intdc)         ***      410d
  DI_DRV(intdiscint)    ***      208d
  GLE_HPIB(gle_hpib_io) 10064d 10072d 10081d 10108d 10119d 10143d 10189d 10190d 10277d 10278d 10511d 10532d
  C_DRV(init_gpio)     ***      226d
  H_DRV(int_hpib)       ***      224d
  IOLIB(general_1)      237d
  IOLIB(general_2)      837d   917d   935d   969d   995d  1014d  1027d  1040d  1053d  1073d
  IOLIB(general_4)      1204d  1249d  1355d  1357d  1441d  1465d  1490d  1533d  1667d
  IOLIB(hpib_0)         1709d  1711d  1713d  1722d  1733d  1744d
  IOLIB(hpib_1)         393d   395d  398d  401d  404d  407d  409d  412d  413d  432d  443d  467d  482d  496d
                        511d  512d  599d  600d  697d  698d  745d  746d
  IOLIB(hpib_2)         1779d  1781d  1785d  1795d  1797d  1800d  1801d  1811d  1839d  1853d  1862d  1889d  1898d  1918d
                        1936d  1952d  1973d  1981d  1995d  2001d
  IOLIB(hpib_3)         2029d  2031d  2037d  2039d  2041d  2043d  2046d  2063d  2080d  2096d  2117d  2135d  2145d  2154d
                        2164d
  IOLIB(serial_0)       2215d  2217d  2219d  2231d  2294d  2357d
  IOLIB(serial_3)       2503d  2506d  2509d  2512d  2515d  2518d  2529d  2598d  2682d  2736d  2801d  2836d  2851d
  KERNEL(general_0)     747d   750d  755d  756d  759d  762d  852d  871d  890d  909d  929d  944d  965d  1253d
  KERNEL(iodeclarations) 368d   616d
  PRINTER(prtdrv)      66d
  RS_DRV(intfs)         219d
  SRM_DRV(srm)          1253d  1297d

type_of_card
  KERNEL(iodeclarations) 381d   541d

```


uep	AMIGO(amigocvr)	428d 513c 544d 587c 624c 675c 710d 777c 839c 903c 956c	439d 515c 545d 596c 632c 678c 720c 776c 873c 906c 958c	470c 516c 546d 598c 634c 680c 734c 789c 874c 910c 953c	472c 519c 547d 598c 634c 681c 737c 793c 880c 914c 960c	473c 521c 548d 604c 645c 682c 752c 801c 882c 917c 961c	474c 524c 557d 605c 646c 685c 753c 801c 888c 921c 964c	475c 525c 563c 606c 648c 686c 755c 802c 893c 922c 966c	476c 527c 566c 607c 654c 689c 756c 802c 893c 922c 966c	477c 528c 567c 609c 659c 690c 758c 802c 893c 922c 966c	478c 529c 568c 611c 661c 691c 758c 802c 893c 922c 966c	483c 530c 568c 613c 666c 693c 762d 802c 893c 922c 966c	490d 533c 581c 614c 667c 693c 766c 802c 893c 922c 966c	511c 535c 582c 615c 668c 693c 768c 802c 893c 922c 966c	512c 543d 583c 618c 671c 700c 770c 837c 896c 943c 966c
	AMIGO(csamigo)	137d 152d 224d 314c 387c 339d 393c 452c 493c	138d 160d 235c 315c 388c 340d 395c 453c 494c	133d 162c 240d 325d 399c 341d 396c 454c 495c	137d 172c 268c 346c 406c 342d 397c 464c 496c	141d 177d 269c 346c 406c 343d 397c 464c 500d	142d 177d 269c 346c 406c 344d 397c 464c 500d	144d 183d 291c 344c 406c 345d 397c 464c 500d	145d 185c 292c 344c 406c 346d 397c 464c 500d	147d 189d 296c 340c 406c 347d 397c 464c 500d	148d 207c 300c 340d 406c 348c 397c 464c 500d	148d 207c 300c 340d 406c 348c 397c 464c 500d	149d 212c 304d 371c 406c 348c 397c 464c 500d	150d 219c 311c 372c 406c 348c 397c 464c 500d	151d 223c 311c 372c 406c 348c 397c 464c 500d
	CS80(cs80)	387c 339d 393c 452c 493c	388c 340d 395c 453c 494c	341d 341d 396c 454c 495c	342d 342d 397c 464c 496c	343d 343d 397c 464c 500d	344d 344d 397c 464c 500d	345d 345d 397c 464c 500d	346d 346d 397c 464c 500d	347d 347d 397c 464c 500d	355d 356d 397c 464c 500d	356d 356d 397c 464c 500d	365d 365d 397c 464c 500d	376c 376c 397c 464c 500d	381d 381d 397c 464c 500d
	CS80(cs80dsr)	531c 602d 761c 910c	552c 604d 763c 923c	556d 608d 763c 944c	579c 611d 763c 945c	580c 620c 763c 950c	589c 621c 763c 951c	589c 622c 763c 951c	589c 622c 763c 951c	589c 622c 763c 951c	650c 650c 763c 951c	650c 650c 763c 951c	701c 701c 763c 951c	702c 702c 763c 951c	757c 757c 763c 951c
	CS80(cs80dvr)	982d 1054d 1129c 1267c 1373c 877d 1051d	993d 1088d 1131c 1279c 1377c 877d 1051d	1002c 1090c 1133c 1282c 1382c 877d 1051d	1003c 1094d 1134c 1293c 1387c 882c 1051d	1004c 1100c 1168d 1307c 1388c 883c 1051d	1005c 1101c 1172c 1307c 1388c 883c 1051d	1006c 1103c 1176c 1307c 1388c 883c 1051d	1007c 1104c 1176c 1307c 1388c 883c 1051d	1009c 1107d 1223c 1353d 1353d	1027d 1114c 1223c 1353d 1353d	1043c 1116c 1230c 1366c 1366c	1047c 1124c 1231c 1366c 1366c	1048c 1126c 1236c 1366c 1366c	1053c 1127c 1250c 1369c 1369c
	CTABLE(ctr)	149c 192d 223c 292c 379c 437d 508c	150c 193d 225c 293c 380c 438d 509c	153d 194d 225c 293c 380c 438d 509c	157c 195d 232c 310d 395c 440c 535c	158c 196d 232c 310d 395c 440c 535c	163d 197d 239c 318c 400c 452c 542c	165c 198d 244c 325c 400c 452c 542c	165c 198d 244c 325c 400c 452c 542c	170c 199d 246c 349d 410c 482c 542c	175d 200d 256d 351c 412c 482c 542c	177c 201d 256d 351c 412c 482c 542c	129c 201d 256d 351c 412c 482c 542c	130c 202d 261c 357d 429c 491c 542c	133c 219d 266c 363c 431c 491c 542c
	DISCHPIB(dischpib)	149c 192d 223c 292c 379c 437d 508c	150c 193d 225c 293c 380c 438d 509c	153d 194d 225c 293c 380c 438d 509c	157c 195d 232c 310d 395c 440c 535c	158c 196d 232c 310d 395c 440c 535c	163d 197d 239c 318c 400c 452c 542c	165c 198d 244c 325c 400c 452c 542c	165c 198d 244c 325c 400c 452c 542c	170c 199d 246c 349d 410c 482c 542c	175d 200d 256d 351c 412c 482c 542c	177c 201d 256d 351c 412c 482c 542c	129c 201d 256d 351c 412c 482c 542c	130c 202d 261c 357d 429c 491c 542c	133c 219d 266c 363c 431c 491c 542c
	F9885(f9885dvr)	78d 393d 559d	122c 466c 560d	131c 467c 561d	136c 481c 562d	137c 482c 563d	148c 489c 564d	164c 492c 565d	201c 507c 567d	254c 514c 569d	255c 514c 569d	260c 514c 569d	279c 586c 586c	286c 590d 590d	286c 612c 612c
	INITLOAD(mini)	393d 559d	466c 560d	467c 561d	481c 562d	482c 563d	489c 564d	492c 565d	507c 567d	514c 569d	514c 569d	514c 569d	586c 586c	590d 590d	612c 612c
	MINIT(cs801r)	616c 676d 757c 841c	617c 692c 763c 842c	613c 693c 763c 843c	620c 696c 766c 847d	621c 697c 766c 847d	625d 701d 790c 866c	643c 701d 790c 866c	644c 722c 800c 866c	645c 723c 800c 866c	649d 724c 800c 866c	666c 728d 813c 866c	670c 750c 814c 866c	671c 754c 818d 866c	672c 755c 835c 866c
	MINIT(hminit)	252d 361c 399c 441c	253d 364c 400c 446c	254d 363c 401c 447c	270d 366c 402c 448c	278c 367c 403c 449c	283c 368c 404c 460c	288c 373c 405c 467c	299d 379c 425c 470c	308c 391c 425c 474c	316c 392c 426c 475c	325c 393c 437c 480c	327c 394c 438c 481c	359c 395c 439c 482c	360c 398c 440c 485c
	MINIT(iramigo)	489c 191d	482c 191d	495c 192d	499c 193d	500c 193d	501c 193d	501c 193d	501c 193d	501c 193d	501c 193d	501c 193d	501c 193d	501c 193d	501c 193d
	MINIT(qminit)	881d 978c 1150c 1198c	881d 978c 1150c 1198c	881d 978c 1150c 1198c	881d 978c 1150c 1198c	881d 978c 1150c 1198c	881d 978c 1150c 1198c	881d 978c 1150c 1198c	881d 978c 1150c 1198c	881d 978c 1150c 1198c	881d 978c 1150c 1198c	881d 978c 1150c 1198c	881d 978c 1150c 1198c	881d 978c 1150c 1198c	881d 978c 1150c 1198c
	MIUI	1267c 19d 482c 1021d 1452c	1268c 35c 483c 1021d 1452c	1271c 35c 483c 1021d 1452c	1275c 37c 483c 1021d 1452c	1276c 37c 483c 1021d 1452c	1277c 167c 483c 1021d 1452c	1277c 167c 483c 1021d 1452c	1277c 167c 483c 1021d 1452c	1277c 167c 483c 1021d 1452c	1277c 167c 483c 1021d 1452c	1277c 167c 483c 1021d 1452c	1277c 167c 483c 1021d 1452c	1277c 167c 483c 1021d 1452c	1277c 167c 483c 1021d 1452c
	TAPEBKUP	482c 1021d 1452c	483c 1021d 1452c	483c 1021d 1452c	483c 1021d 1452c	483c 1021d 1452c	483c 1021d 1452c	483c 1021d 1452c	483c 1021d 1452c	483c 1021d 1452c	483c 1021d 1452c	483c 1021d 1452c	483c 1021d 1452c	483c 1021d 1452c	483c 1021d 1452c
	TAPEBKUP(cs80tbdvr)	309d 352c 393d	313d 353c 395d	313d 353c 401c	320d 357c 402c	322d 358c 407c	323d 364d 411d	324d 366d 413d	325d 366d 419c	329d 373c 420c	329d 373c 425c	329d 373c 425c	329d 373c 425c	329d 373c 425c	329d 373c 425c
	TAPEBKUP(cs80tbr)	442c 651c 925c	443c 651c 929c	443c 651c 931d	452d 661d 937c	472c 670c 938c	493c 671c 943c	509d 671c 949c	530c 671c 954c	531c 671c 954c	531c 671c 954c	531c 671c 954c	531c 671c 954c	531c 671c 954c	531c 671c 954c
	uep1	1005d	1007c	1009c	1010c	1011c	1015d	1017c	1017c	1017c	1017c	1017c	1017c	1017c	1017c
	uep2	1005d	1008c	1009c	1010c	1011c	1015d	1017c	1017c	1017c	1017c	1017c	1017c	1017c	1017c
	uep_proc	1181d	1192c	1192c	1192c	1192c	1192c	1192c	1192c	1192c	1192c	1192c	1192c	1192c	1192c
	uep_proc_type	1181d	1192c	1192c	1192c	1192c	1192c	1192c	1192c	1192c	1192c	1192c	1192c	1192c	1192c
	CTABLE(scanstuff)	1051d	1062d	1062d	1062d	1062d	1062d	1062d	1062d	1062d	1062d	1062d	1062d	1062d	1062d
	TAPEBKUP(cs80tbdvr)	309d	345d	345d	345d	345d	345d	345d	345d	345d	345d	345d	345d	345d	345d
	uep_type	1005d	1015d	1021d	1031d	1061d	1182d	1218d	1427d	1461d	1461d	1461d	1461d	1461d	1461d
	AMIGO(amigodvr)	428d 137d 152d 379d	439d 138d 160d 392d	490d 139d 177d 392d	543d 140d 183d 392d	545d 141d 189d 392d	557d 142d 219d 392d	654d 144d 229d 392d	702c 145d 240d 392d	710d 146d 274d 392d	766c 147d 296d 392d	777c 148d 304d 392d	790d 149d 328d 392d	938c 150d 339d 392d	938c 150d 339d 392d
	AMIGO(csamigo)	137d 152d 224d 314c 387c 339d 393c 452c 493c	138d 160d 235c 315c 388c 340d 395c 453c 494c	133d 162c 240d 325d 399c 341d 396c 454c 495c	137d 172c 268c 346c 406c 342d 397c 464c 496c	141d 177d 269c 346c 406c 343d 397c 464c 500d	142d 177d 269c 346c 406c 344d 397c 464c 500d	144d 183d 291c 344c 406c 345d 397c 464c 500d	145d 185c 292c 344c 406c 346d 397c 464c 500d	147d 189d 296c 340c 406c 347d 397c 464c 500d	148d 207c 300c 340d 406c 348c 397c 464c 500d	148d 207c 300c 340d 406c 348c 397c 464c 500d	149d 212c 304d 371c 406c 348c 397c 464c 500d	150d 219c 311c 372c 406c 348c 397c 464c 500d	151d 223c 311c 372c 406c 348c 397c 464c 500d
	CS80(cs80)	387c 339d 393c 452c 493c	388c 340d 395c 453c 494c	341d 341d 396c 454c 495c	342d 342d 397c 464c 496c	343d 343d 397c 464c 500d	344d 344d 397c 464c 500d	345d 345d 397c 464c 500d	346d 346d 397c 464c 500d	347d 347d 397c 464c 500d	355d 356d 397c 464c 500d	356d 356d 397c 464c 500d	365d 365d 397c 464c 500d	381d 381d 397c 464c 500d	401d 401d 397c 464c 500d
	CS80(cs80dsr)	531c 602d 761c 910c	552c 604d 763c 923c	556d 608d 763c 944c	579c 611d 763c 945c	580c 620c 763c 950c	589c 621c 763c 951c	589c 622c 763c 951c	589c 622c 763c 951c	589c 622c 763c 951c	650c 650c 763c 951c	650c 650c 763c 951c	701c 701c 763c 951c	702c 702c 763c 951c	757c 757c 763c 951c
	CS80(cs80dvr)	982d 1054d 1129c 1267c 1373c 877d 1051d	993d 1088d 1131c 1279c 1377c 877d 1051d	1002c 1090c 1133c 1282c 1382c 877d 1051d	1003c 1094d 1134c 1293c 1387c 882c 1051d	1004c 1100c 1168d 1307c 1388c 883c 1051d	1005c 1101c 1172c 1307c 1388c 883c 1051d	1006c 1103c 1176c 1307c 1388c 883c 1051d	1007c 1104c 1176c 1307c 1388c 883c 1051d	1009c 1107d 1223c 1353d 1353d	1027d 1114c 1223c 1353d 1353d	1043c 1116c 1230c 1366c 1366c	1047c 1124c 1231c 1366c 1366c	1048c 1126c 1236c 1366c 1366c	1053c 1127c 1250c 1369c 1369c
	CS80(tapebuf)	48d	48d	48d	48d	48d	48d	48d	48d	48d	48d	48d	48d	48d	48d
	CTABLE(scanstuff)	1049d	1051d	1052d	1053d	1053d	1053d	1053d	1053d	1053d	1053d	1053d	1053d	1053d	1053d
	DISCHPIB(bkgn)	35d	62d	63d	64d	65d	66d	67d	105d	125d	146d	155d	163d	175d	237d

unblockedddam															
INIT (initunits)		2164d	2244c	2246c	2250c										
INIT (misc)		213d	426d												
LIFDAM (lifmodule)	***	1159c													
UCSD_DAM (ucsdmodule)	***	650c													
unbuf_read_cmd															
AMIGO (amigodvr)	***	732c													
AMIGO (csamigo)	***	53d													
unbuf_write_cmd															
AMIGO (amigodvr)	***	733c													
AMIGO (csamigo)	***	55d													
unbuffereddam															
INIT (misc)		483d	706c												
unclipped_draw															
GLE_HPGL (gle_hppl_out)	***	7526c													
GLE_TYPES (gle_tyes)	***	1181d													
unclipped_move															
GLE_HPGL (gle_hppl_out)	***	7526c													
GLE_TYPES (gle_tyes)	***	1180d													
unconvertible_data_error															
AMIGO (amigodvr)		807c	872c												
AMIGO (csamigo)	***	73d													
MINIT (hminit)	***	318c													
undefined															
CS80 (cs80dsr)	***	629c													
CS80 (cs80dvr)		1156c	1172c	1178c	1186c	1306c	1369c								
CS80 (tapebuf)		42d	57c												
TAPEBKUP		1245c	1482c												
TAPEBKUP (cs80tdvr)		583c	815c												
under_lock_do															
TAPEBKUP		1181d	1239c	1247c	1248c	1266c	1269c	1286c	1369c	1372c	1417c	1449c	1452c	1483c	1486c
uninitialized_media															
CS80 (cs80)	***	166d													
CS80 (cs80dsr)	***	681c													
TAPEBKUP (cs80tdvr)		561c	793c												
unit															
AMIGO (csamigo)		64c	102d	268c											
CS80 (cs80)		340d	343d	416d	427c	433d	439c	473d	479c						
ETU		222d	235c	355d											
INIT (misc)		212d	257d	267c	268c	269c	280c								
MINIT (cs80ir)		660d	625d	639c											
TAPEBKUP (cs80tdvr)		713d	719c	721c	724c	730c	740c	742c	745c						
TAPEBKUP (cs80tbr)		24d	55d	65c											
unit_available															
AMIGO (csamigo)	***	70d													
unit_busy															
AMIGO (amigodvr)	***	597c													
CS80 (cs80dvr)	***	1373c													
DISCHP1B (bkgn)		66d	163d	166c	169c	177c									
unit_fault															
CS80 (cs80)	***	154d													
CS80 (cs80dsr)	***	669c													
TAPEBKUP (cs80tdvr)		548c	781c												
unit_missing															
CTABLE		1342d	1347c	1349c	1350c										
unit_num															
SRR_IRV (srm)		262d	1504c												
unit_number															
SwVol		8d	13c	14c	18c	19c									
unit_prefix_successful															
CTABLE		1229d	1235c	1801c	1806c	1807c									
unit_unavailable															
AMIGO (amigodvr)	***	879c													
AMIGO (csamigo)	***	80c													

II-174

[illegible]

[illegible]

II-180

[illegible]

[illegible]

zbadana										
AMIGO(amigodvr)	***	605c								
CS80(cs80dvr)	***	1389c								
F9885(f9885dvr)	***	119c								
INIT(misc)	***	326c								
INITLOAD(syglotals)	***	188d								
zbadhardware										
AMIGO(amigodvr)		889c	914c							
BUBBLES(bubble)		41c	42c							
CS80(cs80dsr)		651c	671c							
DISCHPAB(dischpab)	***	353c								
F9885(f9885dvr)	***	242c								
INIT(misc)	***	324c								
INITLOAD(bootdammodule)	***	601c								
INITLOAD(mini)		344d	352d	353d	354d	355d	356d	369d		
INITLOAD(syglotals)	***	188d								
INIT(cs801r)		761c	808c							
INIT(hminit)		303c	475c							
MIUI	***	211c								
TAPEBKUP(cs80tbdvr)		531c	551c	759c	783c					
TAPEBKUP(cs80tbr)	***	286c								
zbadmode										
AMIGO(amigodvr)	***	609c								
BUBBLES(bubble)	***	45c								
CS80(cs80dsr)	***	679c								
CS80(cs80dvr)	***	1249c								
EPR0MS(eproms)	***	111c								
F9885(f9885dvr)	***	283c								
INIT(misc)	***	307c								
INITLOAD(bootdammodule)		760c	768c							
INITLOAD(mini)	***	511c								
INITLOAD(syglotals)	***	184d								
INIT(qminit)		986c	1134c							
PRINTER(prtdvr)	***	212c								
TAPEBKUP(cs80tbdvr)		555c	787c							
zbit										
CS80(cs80)		507d	514c							
zcatchall										
AMIGO(amigodvr)		477c	528c	802c	906c	910c	926c	931c	943c	
AMIGO(csamigo)	***	212c								
BUBBLES(bubble)	***	49c								
CS80(cs80dsr)		744c	756c							
CS80(cs80dvr)	***	1129c								
F9885(f9885dvr)		132c	156c	186c	202c	209c	225c	237c	245c	
INIT(misc)	***	325c								
INITLOAD(bootdammodule)	***	602c								
INITLOAD(mini)	***	368d								
INITLOAD(syglotals)	***	188d								
INIT(hminit)	***	492c								
INIT(qminit)	***	1137c								
TAPEBKUP(cs80tbdvr)		626c	646c	857c	879c					
TAPEBKUP(cs80tbr)	***	294c								
zchr										
KEYS(keys)		76d	87c							
zdata										
HOUSE(mouse)	***	48d								
zero										
M68KSYS(ci)	***	773d								
MIUI		402d	437d							
zero_fields										
CTABLE		1243d	1253c	1255c	1257c	1259c	1261c			
zero_out_na_fields										
CTABLE		1239d	1392c	1485c						
zero_parity										
IO1B(serial_3)		2753c	2775c							
KERNEL(iodeclarations)	***	404d								
zerodate										
SRMAM(srmdamodule)		387d	411c	413c						
zerodate_time										
LIFUAM(lifmodule)		585d	616c							
zeromem										
INIT(ldr)	***	2472c								
INITLOAD(loader)		1089d	1567d	1635c						
SEGMENTER(asm)	***	208c								
zerotime										
SRMAM(srmdamodule)		388d	412c	414c						
zinitail										
CS80(cs80dsr)	***	686c								
INIT(misc)	***	321c								
INITLOAD(mini)		342d	343d							
INITLOAD(syglotals)	***	187d								
INIT(hminit)		325c	327c	361c	379c	395c	400c	405c	426c	449c
INIT(qminit)	***	1254c								
TAPEBKUP(cs80tbdvr)		566c	798c							
zit1										
SRM DRV(srm)	***	297d								
zit2										
SKM DRV(srm)	***	298d								
zmediumchanged										
AMIGO(amigodvr)		607c	668c	903c						
CS80(cs80dsr)	***	703c								
CS80(cs80dvr)	***	1391c								
F9885(f9885dvr)		138c	280c							
INIT(misc)	***	355c								
INITLOAD(mini)		349d	464c	508c						
INITLOAD(syglotals)	***	195d								
LIFUAM(lifmodule)	***	214c								
INIT(hminit)		474c	480c							
TAPEBKUP(cs80tbdvr)		585c	817c							
znoblock										
AMIGO(amigodvr)		828c	874c	923c						
CS80(cs80dsr)	***	693c								
F9885(f9885dvr)		228c	235c							
INIT(misc)	***	336c								
INITLOAD(bootdammodule)	***	758c								
INITLOAD(mini)		351d	357d	360d	361d	362d	363d			
INITLOAD(syglotals)	***	191d								
MIUI	***	120d								
TAPEBKUP	***	1051d								
TAPEBKUP(cs80tbdvr)		573c	805c							

znodevice												
AMIGO(amigodvr)		470c	478c	483c	516c	519c	529c	533c	535c	567c	880c	892c
AMIGO(csam19g)	***	172c										
BUBBLES(bubble)		46c	64c									
CS80(cs80dsr)		677c	923c									
CS80(cs80dvr)		1005c	1104c	1360c								
DISCHPIB(dischpib)		222c	249c									
EPROMS(eproms)		47c	102c									
F9885(f9885dvr)		89c	217c	255c	262c							
INIT(initunits)		2230c	2233c									
INIT(misc)	***	320c										
INITLOAD(bootdammodule)	***	597c										
INITLOAD(mini)		370d	483c	490c	493c							
INITLOAD(sysglobals)	***	187d										
MINIT(qminit)	***	958c										
MIUI	***	210c										
PRINTER(prtdvr)		105c	140c	290c								
SRMAM(srmammodule)		151c	159c	189c								
SRMAM(srmammodule)	***	1645c										
SRM_DRV(srm)		1152c	1470c									
TAPEBKUP(cs80tbdvr)		351c	557c	789c								
znomedium												
AMIGO(amigodvr)	***	893c										
CS80(cs80dvr)	***	1238c										
F9885(f9885dvr)	***	220c										
INIT(misc)	***	338c										
INITLOAD(bootdammodule)	***	598c										
INITLOAD(mini)		345d	346d	347d	348d	350d	367d					
INITLOAD(sysglobals)	***	191d										
MINIT(hminit)	***	481c										
znosuchblk												
AMIGO(amigodvr)	***	922c										
BUBBLES(bubble)	***	44c										
CS80(cs80dsr)	***	680c										
EPROMS(eproms)	***	54c										
INIT(misc)	***	319c										
INITLOAD(sysglobals)	***	187d										
TAPEBKUP(cs80tbdvr)		560c	792c									
znotready												
AMIGO(amigodvr)	***	891c										
BUBBLES(bubble)	***	71c	88c									
CS80(cs80dsr)	***	688c										
INIT(misc)	***	337c										
INITLOAD(bootdammodule)	***	599c										
INITLOAD(sysglobals)	***	191d										
SRM_DRV(srm)	***	1181c										
TAPEBKUP		1401c	1403c									
TAPEBKUP(cs80tbdvr)		568c	800c									
zprotected												
AMIGO(amigodvr)		882c	897c									
CS80(cs80dsr)	***	690c										
EPROMS(eproms)	***	76c										
F9885(f9885dvr)	***	224c										
INIT(misc)	***	322c										
INITLOAD(mini)		358d	358d									
INITLOAD(sysglobals)	***	187d										
MINIT(hminit)	***	482c										
TAPEBKUP(cs80tbdvr)		570c	802c									
zstrange												
BUBBLES(bubble)	***	43c										
INIT(misc)	***	323c										
INITLOAD(mini)		365d	366d									
INITLOAD(sysglobals)	***	188d										
ztimeout												
BUBBLES(bubble)	***	48c										
DISCHPIB(dischpib)	***	225c										
INIT(misc)	***	308c										
INITLOAD(sysglobals)	***	184d										
PRINTER(prtdvr)		132c	199c	316c								
SRM_DRV(srm)	***	1366c										
zunitialized												
AMIGO(amigodvr)		671c	899c									
CS80(cs80dsr)	***	684c										
CS80(cs80dvr)	***	1242c										
INIT(misc)	***	335c										
INITLOAD(sysglobals)	***	191d										
MIUI	***	121d										
TAPEBKUP	***	1052d										

A804XDVR

Description

A804XDVR contains the routines for talking to the 8041 and 8042 keyboard processors, as well as the procedure through which all interrupts are funneled and sent to the appropriate places.

Requirements

SYSGLOBALS, SYSDEVS, ISR, and ASM.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1: (*
2:
3: (c) Copyright Hewlett-Packard Company, 1983.
4: All rights are reserved. Copying or other
5: reproduction of this program except for archival
6: purposes is prohibited without the prior
7: written consent of Hewlett-Packard Company.
8:
9:
10: RESTRICTED RIGHTS LEGEND
11:
12: Use, duplication, or disclosure by the Government
13: is subject to restrictions as set forth in
14: paragraph (b) (3) (B) of the Rights in Technical
15: Data and Computer Software clause in
16: DAR 7-104.9(a).
17:
18: HEWLETT-PACKARD COMPANY
19: 0 Fort Collins, Colorado *)
20:
21: 0 $SYSPROGS
22: 0 $heap_dispose off$
23: 0 $iocheck off$
24: 0 $range off$ $ovflcheck off$
25: 0 $debug off$
26: 0 $stackcheck off$
27:
28: 0 PROGRAM A804XINIT;
29: 1 MODULE A804XDVR;
30: 1 $SEARCH 'INITLOAD', 'ASM', 'SYSDEVS', 'INIT'S
31: 1 IMPORT SYSGLOBALS, SYSDEVS, ISR, ASM;
32: 1 EXPORT
33: 1 TYPE
34: 1 DATAHOOKTYPE = PROCEDURE (DPTA:BYTE);
35: 1 VAR
36: -8 1 DATAHOOK : DATAHOOKTYPE;
37: -16 1 STATUSSHOOK : KBDHOOKTYPE;
38: -24 1 STATUS$HOOK : KBDHOOKTYPE;
39:
40: -24 1 PROCEDURE SENDCMD(CMD:BYTE);
41: -24 1 PROCEDURE SENDDATA(DATA:BYTE);
42: -24 1 PROCEDURE CMD_READ_1(CMD:BYTE; VAR DATA:BYTE);
43:
44: -24 1 FUNCTION INITA804X:BOOLEAN;
45:
46: -24 1 IMPLEMENT
47:
48: -24 1 CONST
49: -24 1 UP = TRUE; DOWN = FALSE;
50: -24 1 TYPE
51: -24 1 STATUSTYPE = PACKED RECORD
52: -24 1 CASE INTEGER OF
53: -24 1 0:(PAD1 :0..63;
54: -24 1 BUSY :BOOLEAN;
55: -24 1 READY :BOOLEAN);
56: -24 1 1:(STATBYTE :CHAR);
57: -24 1 END;
58: -24 1 STRING3 = STRING[3];
59:
60: -24 1 VAR

```

```

61: -24 1 DATPREG [HEX('428001')] : CHAR;
62: -24 1 STATUSREG [HEX('428003')] : CHAR;
63: -24 1 CHDREG [HEX('428003')] : CHAR;
64: -44 1 ISRREC : ISRIB;
65: -46 1 MAXDATA : SHORTINT;
66: -48 1 HAVEDATA : SHORTINT;
67: -52 1 DATABUFFER : WINDOWP;
68: -54 1 EXTLEFT, EXTRIGHT : BOOLEAN;
69: -56 1 MASK : BYTE;
70:
71: 1 PROCEDURE DUMMYSSTATUS$6(VAR STATBYTE,DATABYTE:BYTE; VAR DONE:BOOLEAN);
72: 2 BEGIN END;
73:
74: 1 PROCEDURE STATUSISR(VAR STATBYTE,DATABYTE:BYTE; VAR DONE:BOOLEAN);
75: 2 BEGIN ( IF ANYTHING WRONG THEN RESET EXTENDCHPR 8ITS )
76: 2 ( OTHERWISE IGNORE IT )
77: 2 IF DATABYTE>127 THEN BEGIN EXTLEFT:=FALSE; EXTRIGHT:=FALSE; END;
78: 2 END;
79:
80: 1 PROCEDURE DATAISR(DATA:BYTE);
81: 2 BEGIN
82: 2 IF HAVEDATA<MPXDATA THEN
83: 3 BEGIN DATABUFFER[HAVEDATA]:=CHR(DATA);
84: 3 HAVEDATA:=HAVEDATA+1;
85: 3 END;
86: 2 END;
87:
88: 1 PROCEDURE A804XISR(ISRIBPTR : ISRIB);
89: 2 VAR
90: 2 STATUS,DATA : BYTE;
91: 2 DOIT : BOOLEAN;
92: -4 -5 BEGIN
93: 2 STATUS := ORD(STATUSREG); { READ STATUS REG }
94: 2 DATA := ORD(DATAREG); { READ DATA REG }
95: 2 DOIT := TRUE;
96: 2 CASE STATUS DIV 16 OF
97: 2 0,7 : { UN IMPLEMENTED OPERATIONS };
98: 2 1,2,3 : CALL(TIMERISRHOOK,STATUS,DATA,DOIT);
99: 2 4 : CALL(DATAHOOK,DATA); { REQUESTED DATA OTHER THAN CARAVAN }
100: 2 5 : CALL(STATUSSHOOK,STATUS,DATA,DOIT); { CARAVAN STATUS CHANGE }
101: 2 6 : CALL(STATUS$HOOK,STATUS,DATA,DOIT); { CARAVAN RAW DATA }
102: 2 8..11 : BEGIN
103: 3 CALL(KBDTRANSHOOK,STATUS,DATA,DOIT);
104: 3 IF DOIT THEN CALL(KBDISRHOOK,STATUS,DATA,DOIT);
105: 3 END;
106: 3 OTHERWISE CALL(RPGISRHOOK,STATUS,DPTA,DOIT);
107: 3 END;
108: 2 END;
109:
110: 1 PROCEDURE POLLISR(WAIT:BOOLEAN);
111: 2 VAR KBDSTATUS : STATUSTYPE;
112: -2 -6 ISRIBPTR : ISRIB;
113: 2 BEGIN
114: 2 IF INTLEVEL>0 THEN
115: 3 BEGIN ( POLL FOR INTERRUPT OR JUST LEAVE )
116: 3 REPEAT KBDSTATUS,STATBYTE:=STATUSREG;
117: 3 UNTIL KBDSTATUS.READY OR NOT WAIT;
118: 4

```



```

121:C      3      IF KBDSTATUS.READY THEN
122:C      4      BEGIN ISIRIBPTR := ADDR(ISRREC); AB04XISR(ISRIBPTR); END;
123:C      3      END;
124:C      2      END;
125:S
126:D      1      PROCEDURE SETUPREAD(COUNT:SHORTINT; ANYVAR BUFFER:WINDOW; OFFSET:SHORTINT);
127:C      2      BEGIN
128:C      2      MAXDATA:=COUNT; HAVEDATA:=0; DATAEUFFER:=ADDR(BUFFER,OFFSET);
129:C      2      END;
130:S
131:D      1      PROCEDURE WAIT4KBDREADY;
132:D      2      VAR KBDSTATUS : STATUSTYPE;
133:C      2      BEGIN
134:C      2      REPEAT KBDSTATUS.STATBYTE:=STATUSREG;
135:C      3      UNTIL NOT KBDSTATUS.BUSY;
136:C      3      END;
137:S
138:D      1      PROCEDURE SENDCMD(CMD:BYTE);
139:C      2      BEGIN WAIT4KBDREADY; CMDREG:=CHR(CMD);
140:C      2      END;
141:S
142:D      1      PROCEDURE SENDDATA(DATA:BYTE);
143:C      2      BEGIN WAIT4KBDREADY; DATAREG:=CHR(DATA);
144:C      2      END;
145:S
146:D      1      PROCEDURE CMD_READ_1(CMD:BYTE; VAR DATA:BYTE);
147:C      2      BEGIN SETUPREAD(1,DATA,1); SENDCMD(CMD);
148:C      2      WHILE HAVEDATA<MAXDATA DO POLLISR(TRUE);
149:C      2      END;
150:S
151:D      1      PROCEDURE SENDBYTESLSF(N:SHORTINT; ANYVAR BUFFER:WINDOW; OFFSET:SHORTINT);
152:D      2      VAR I : SHORTINT;      { SEND LEAST SIGNIFICANT BYTE FIRST }
153:C      2      BEGIN
154:C      2      FOR I:=N-1 DOWNT0 0 DO SENDDATA(ORD(BUFFER[I+OFFSET]));
155:C      2      END;
156:S
157:D      1      PROCEDURE BEEPDP(FREQUENCY,DURATION : BYTE);
158:C      2      BEGIN
159:C      2      SENDCMD(HEX('A3'));
160:C      2      SENDDATA((256-DURATION) MOD 256); SENDDATA(FREQUENCY);
161:C      2      END;
162:S
163:D      1      PROCEDURE MASKOPS(ENABLE,DISABLE:BYTE);
164:C      2      BEGIN
165:C      2      MASK:=IOR(DISABLE,IAND(MASK,-1-ENABLE));
166:C      2      SENDCMD(HEX('40')+MASK);
167:C      2      END;
168:S
169:D      1      PROCEDURE DO_RPGOPS(CMD : BYTE; VAR DATA : BYTE);
170:C      2      BEGIN
171:C      2      CASE CMD OF
172:C      3      0: MASKOPS(KBDMASK,0);      { ENABLE RPG }
173:C      3      1: MASKOPS(0,KBDMASK);      { DISABLE RPG }
174:C      3      2: BEGIN SENDCMD(HEX('A6')); SENDDATA(DATA); END;      { SET RATE }
175:C      3      3: BEGIN SENDCMD(HEX('26')); CMD_READ_1(HEX('17'),DATA); END;      { FLX 4/12/84 SFB GET RATE }
176:C      3      OTHERWISE
177:C      3      OTHERWISE
178:C      3      END;
179:C      2      END;
180:S

```

```

181:D      1      PROCEDURE DO_KBDOPS(CMD: BYTE; VAR DATA: BYTE);
182:D      2      TYPE
183:D      2      LANGITF=ARRAY[0..31] OF LANGTYPE;
184:D      2      LANGNONITF=ARRAY[0..5] OF LANGTYPE;
185:D      2      CONST
186:D      2      WHICHITFLANG=LANGITF[ND_KBD,ND_KBD,ND_KBD,SWISS_FR_KBD,NO_KBD,NO_KBD,
187:D      2      NO_KBD,CON_ENG_KBD,NO_KBD,ND_KBD,ND_KBD,ITALIAN_KBD,
188:D      2      NO_KBD,DUTCH_KBD,SWEDISH_KBD,GERMAN_KBD,
189:D      2      NO_KBD,NO_KBD,NO_KBD,SPANISH_EUR_KBD,NO_KBD,
190:D      2      BELGIAN_KBD,FINISH_KBD,UK_KBD,CON_FR_KBD,
191:D      2      SWISS_GR_KBD,NORWEGIAN_KBD,FRENCH_KBD,DANISH_KBD,
192:D      2      KATAKANA_KBD,SPANISH_LATIN_KBD,US_KBD];
193:D      2      WHICHNONITFLANG = LANGNONITF[US_KBD,FRENCH_KBD,GERMAN_KBD,SWEDISH_KBD,
194:D      2      SPANISH_KBD,KATAKANA_KBD];
195:D      2      VAR C:BYTE;
196:C      2      BEGIN
197:C      2      CASE CMD OF
198:C      3      0: MASKOPS(KBDMASK,0);      { ENABLE KEYBOARD }
199:C      3      1: MASKOPS(0,KBDMASK);      { DISABLE KEYBOARD }
200:C      3      2,3: BEGIN
201:C      4      IF CMD=2 THEN C:=HEX('A0'); { SET AUTO DELAY }
202:C      4      ELSE C:=HEX('A2'); { SET REPEAT RATE }
203:C      4      SENDCMD(C); SENDDATA(256-DATA);
204:C      4      END;
205:C      3      4,5: BEGIN
206:C      4      IF CMD=4 THEN C:=HEX('20'); { GET DELAY }
207:C      4      ELSE C:=HEX('22'); { GET REPEAT }
208:C      4      SENDCMD(C);      { COPY DATA TO TIMER - SFB 3/20/84 }
209:C      4      CMD_READ_1(HEX('17'),DATA); DATA:=256-DATA; { READ byte from TIMER
210:C      4      SFB 3/20/84 }
211:C      4      END;
212:C      3      6: BEGIN      { SET KBDTYPE }
213:C      4      CMD_READ_1(HEX('11'),KBDCONFIG);
214:C      4      IF ODD(KBDCONFIG DIV 32) THEN KBDTYPE:=ITFKBD
215:C      4      ELSE
216:C      4      IF ODD(KBDCONFIG) THEN KBDTYPE:=SMALLKBD
217:C      4      ELSE KBDTYPE:=LARGEKBD;
218:C      4      DATA:=KBDCONFIG;
219:C      4      END;
220:C      3      7: BEGIN      { SET KBDLANG }
221:C      4      CMD_READ_1(HEX('12'),DATA);
222:C      4      CASE KBDTYPE OF
223:C      5      SMALLKBD,LARGEKBD : KBDLANG:=WHICHNONITFLANG[DATA];
224:C      5      ITFKBD : KBDLANG:=WHICHITFLANG[DATA MOD 32];
225:C      5      OTHERWISE
226:C      5      KBDLANG:=ND_KBD;
227:C      5      END;
228:C      4      OTHERWISE
229:C      4      OTHERWISE
230:C      4      END;
231:C      3      END;
232:D      1      PROCEDURE SEND_WAIT(L:STRING3);
233:D      2      VAR I,J : SHORTINT;      { SEND COMMANDS & WAIT FOR DATA }
234:C      2      BEGIN
235:C      2      FOR I:=1 TO STRLEN(L) DO
236:C      3      BEGIN J:=HAVEDATA; SENDCMD(ORD(L[I])); WHILE J=HAVEDATA DO; END;
237:C      3      END;
238:S
239:D      1      PROCEDURE CLOCKOPS(CMD:CLOCKOP; VAR THETIME:RTIME);
240:C      2      BEGIN

```



```

241:C      IF CMD=CGET THEN
242:C      BEGIN (CGET) { READ THE TIME AND DATE }
243:C      THETIME.PACKEDTIME:=0; SETUPREAD(3,THETIME.PACKEDTIME,1);
244:C      SENDCMD(HEX('31')); SEND_WAIT(#21#20#19);
245:C      THETIME.PACKEDDATE:=0; SETUPREAD(2,THETIME.PACKEDDATE,2);
246:C      SEND_WAIT(#23#22);
247:C      END
248:C      ELSE
249:C      BEGIN (CFUT) { SET THE TIME AND DATE }
250:C      SENDCMD(HEX('AD')); SENDBYTESLSF(3,THETIME.PACKEDTIME,1);
251:C      SENDCMD(HEX('AP')); SENDBYTESLSF(2,THETIME.PACKEDDATE,2);
252:C      END;
253:C      END;
254:S
255:D      1  PROCEDURE TIMEROPS(TIMER:TIMERTYPES; OP:TIMEROPTYPE;
256:D      2  VAR TDATA:INTEGER; VAR TD:TIMERDATA);
257:D      3  C      : BYTE;
258:D      4  BEGIN
259:D      5  CASE DP OF
260:D      6  SETT:
261:D      7  CASE TIMER OF
262:D      8  DELAYT,CYCLICT:
263:D      9  BEGIN
264:D      10 IF TIMER=CYCLICT THEN C:=HEX('BA') ELSE C:=HEX('B7');
265:D      11 SENDCMD(C);
266:D      12 IF TD.COUNT=0 THEN SENDCMD(HEX('31')) { TO CANCEL }
267:D      13 ELSE
268:D      14 BEGIN TDATA:=16777216-TD.COUNT; SENDBYTESLSF(3,TDATA,1); END;
269:D      15 END;
270:D      16 PERIODICT: { DON'T DO ANY THING }
271:D      17 DELAY7T: BEGIN
272:D      18 SENDCMD(HEX('B2'));
273:D      19 IF TD.COUNT=0 THEN SENDCMD(HEX('31')) { TO CANCEL }
274:D      20 ELSE
275:D      21 BEGIN TDATA:=65536-TD.COUNT; SENDBYTESLSF(2,TDATA,2); END;
276:D      22 END;
277:D      23 MATCHT: BEGIN
278:D      24 TDATA:=(TD.MATCH.HOUR*360000)+(TD.MATCH.MINUTE*6000)+
279:D      25 TD.MATCH.CENTISECOND;
280:D      26 SENDCMD(HEX('B4'));
281:D      27 IF TDATA=0 THEN SENDCMD(HEX('31')) { TO CANCEL }
282:D      28 ELSE SENDBYTESLSF(3,TDATA,1);
283:D      29 END;
284:D      30 OTHERWISE
285:D      31 END; { CASE TIMER }
286:D      32 READT:
287:D      33 BEGIN
288:D      34 TDATA:=0;
289:D      35 CASE TIMER OF
290:D      36 DELAYT,CYCLICT:
291:D      37 BEGIN
292:D      38 SETUPREAD(3,TDATA,1);
293:D      39 IF TIMER=CYCLICT THEN C:=HEX('3E') ELSE C:=HEX('3B');
294:D      40 SENDCMD(C); SEND_WAIT(#21#20#19);
295:D      41 TD.COUNT:=16777216-TDATA;
296:D      42 END;
297:D      43 PERIODICT: TD.COUNT:=1;
298:D      44 DELAY7T: BEGIN
299:D      45 SETUPREAD(2,TDATA,2);
300:D      46

```

```

301:C      SENDCMD(HEX('36')); SEND_WAIT(#20#19);
302:C      TD.COUNT:= 65536-TDATA;
303:C      END;
304:C      MATCHT: BEGIN
305:C      1  SETUPREAD(3,TDATA,1);
306:C      2  SENDCMD(HEX('38')); SEND_WAIT(#21#20#19);
307:C      3  TD.MATCH.HOUR:=TDATA DIV 360000;
308:C      4  TD.MATCH.MINUTE:=TDATA-(TD.MATCH.HOUR*360000) DIV 6000;
309:C      5  TD.MATCH.CENTISECOND:= TDATA MOD 6000;
310:C      6  END;
311:C      OTHERWISE
312:C      7  END; { CASE TIMER }
313:C      8  END;
314:C      GETINFO:
315:C      9  BEGIN
316:C      10 TD.RESOLUTION:=10000;
317:C      11 IF TIMER=DELAY7T THEN TD.RANGE:=65535
318:C      12 ELSE
319:C      13 IF TIMER=PERIODICT THEN TD.RANGE:=1
320:C      14 ELSE TD.RANGE:=16777215;
321:C      15 END;
322:C      16 END; { CASE DP }
323:C      17 END;
324:S
325:S
326:D      1  PROCEDURE KEYTRANS(VAR STATBYTE; KEY: BYTE; VAR DOKEY: BOOLEAN);
327:D      2  VAR EXTSTATE: BOOLEAN;
328:D      3
329:D      4  PROCEDURE SSET(VAR K:BOOLEAN);
330:D      5  BEGIN K:=TRUE; DOKEY:=TRANSMODE=KPASS_EXTC;
331:D      6  EXTSTATE:= DOKEY;
332:D      7  END;
333:D      8  PROCEDURE SCLEAR(VAR K:BOOLEAN);
334:D      9  BEGIN K:=FALSE; DOKEY:=FALSE;
335:D      10 IF TRANSMODE=KPASS_EXTC THEN EXTSTATE:= TRUE
336:D      11 ELSE EXTSTATE:= NOT (EXTLEFT OR EXTRIGHT);
337:D      12 END;
338:D      13
339:D      14 BEGIN (KEYTRANS)
340:D      15 STATBYTE := (STATBYTE DIV 16)*16; DOKEY:=TRUE;
341:D      16 IF KBDTYPE=ITFKBD THEN
342:D      17 BEGIN
343:D      18 IF TRANSMODE=KPASSSTHRU THEN EXTSTATE:= UP
344:D      19 ELSE
345:D      20 BEGIN { NOT PASSSTHRU }
346:D      21 IF KEY=18 THEN SSET(EXTLEFT)
347:D      22 ELSE
348:D      23 IF KEY=19 THEN SSET(EXTRIGHT)
349:D      24 ELSE
350:D      25 IF KEY=146 THEN SCLEAR(EXTLEFT)
351:D      26 ELSE
352:D      27 IF KEY=147 THEN SCLEAR(EXTRIGHT)
353:D      28 ELSE
354:D      29 BEGIN
355:D      30 IF KBDSSYSMODE THEN
356:D      31 CASE KEY OF
357:D      32 27: KEY:=26; {F1=K0}
358:D      33 28: KEY:=42; {F2=RECALL}
359:D      34 29: KEY:=51; {F5=STEP}
360:D      35 30: KEY:=49; {F6=ALPHA}

```



```

361:C      10      31: KEY:=S0; {F7=GRAPHICS}
362:C      10      32: KEY:=4S; {F3=CLR->END}
363:C      10      33: KEY:=S8; {F4=CONTINUE}
364:C      10      34, 35: ; { UP AND DOWN ARROW KEYS }
365:C      10      36: KEY:=37; {F8=K9}
366:C      10      OTHERWISE
367:C      10      END; { CASE KEY }
368:C      8        IF TRANSMODE=KSHIFT_EXTC THEN EXTSTATE:= NOT(EXTLEFT OR EXTRIGHT)
369:C      9        ELSE EXTSTATE:= UP;
370:C      8        END;
371:C      4        END; { NOT PASSTHRU MODE }
372:C      3        STATBYTE:=STATBYTE+(ORD(EXTSTATE)*8)+7; { INCLUDE EXTCHAR STATUS }
373:C      3        END { ITFKBO }
374:C      3        ELSE STATBYTE:=STATBYTE+1S; { TURN ON ALL LOW ORDER BITS }
375:C      2        END; { KEYTRANS }
376:S
377:D      1        FUNCTION INITA804X:BOOLEAN;
378:D      2        VAR
379:D      -2      2        TEMP : BYTE;
380:C      2        BEGIN
381:C      2        INITA804X:=FALSE;
382:C      2        TRY
383:C      3        TEMP := ORD(STATUSREG); { IS THE 8041/8042 PRESENT ? }
384:C      3        MASK := 0; { INITIALIZE MASK }
385:C      3        MASKOPS(0,KBOMASK+RESETMASK+TIMERMASK+PSIMASK+FHIMASK);
386:C      3        PERMISSRLINK(A804XISR,ADDR(STATUSREG),1,1,1,ADDR(ISRREC));
387:C      3        HAVEQDATA := 0; MAXQDATA := 0;
388:C      3        BFREQUENCY:=8; BOURATION:=8;
389:C      3        BEEPERHOOK := BEEPOP;
390:C      3        RPDREQHOOK := DO_RPDOPS;
391:C      3        KBDREQHOOK := DO_KBDOPS;
392:C      3        KBDPOLLHOOK:= POLLISR;
393:C      3        CLOCKIOHOOK:= CLOCKOPS;
394:C      3        TIMERIOHOOK:= TIMEROPS;
395:C      3        ODATAHOOK := DATAISR;
396:C      3        DO_KBDOPS(SET_KBDTYPE,TEMP);
397:C      3        DO_KBDOPS(SET_KBOLANG,TEMP);
398:C      3        KBDYSMMODE :=TRUE;
399:C      3        IF KBDTYPE=ITFKBD THEN SETSTATUS(6,'S');
400:C      3        TRANSMODE := KPASSTHRU;
401:C      3        KBDALTLOCK := FALSE; KBDSCAPLOCK:=TRUE;
402:C      3        EXTLEFT := FALSE; EXTRIGHT := FALSE;
403:C      3        KBDTRANSHOOK:= KEYTRANS;
404:C      3        MASKOPSHOOK := MASKOPS;
405:C      3        STATUS5HOOK := DUMMYSSTATUS56;
406:C      3        STATUS6HOOK := DUMMYSSTATUS56;
407:C      3        INITA804X := TRUE;
408:C      3        RECOVER
409:C      2        IF ESCAPECODE<>-12 THEN ESCAPE(ESCAPECODE);
410:C      2        END; { INITA804X }
411:S
412:C      1        ENO; {MODULE}
413:S
414:D      1        IMPORT A804XDVR,LOADER;
415:C      1        BEGIN
416:C      1        IF INITA804X THEN MARKUSER;
417:C      1        END. {PROGRAM}

```

No errors. No warnings.

***** Nonstandard language features enabled *****

AMIGO

Description

AMIGO is the transfer method (driver) for all supported Amigo family discs.

Usage

AMIGO supports the following discs:

- 8290x
- 9121
- 9895
- 913x

Requirements

DISCHPIB, DRVASM, IODECLARATIONS, HPIB or DISC_INTF

Optionally: DMA

Notes

AMIGO does not support the 7905, 7906, 7920 and 7925 hard discs. These discs may not be used with Series 200 computers.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*)
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1983.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S     RESTRICTED RIGHTS LEGEND
11:S
12:S     Use, duplication, or disclosure by the Government
13:S     is subject to restrictions as set forth in
14:S     paragraph (b) (3) (B) of the Rights in Technical
15:S     Data and Computer Software clause in
16:S     DAR 7-104.9(a).
17:S
18:S     HEWLETT-PACKARD COMPANY
19:S     0 Fort Collins, Colorado      *)
20:S
21:S
22:D     0 $modcal$
23:D     0 $debug off, range off, ovflcheck off$
24:D     0 $stackcheck off, iocheck off$
25:S
26:D     0 $search 'DRVASM', 'DISCHPIB', 'IOLIB:KERNEL'$
27:S

```

```

28:D     0 $page$
29:D     0 $copyright 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'$
30:S
31:S
32:D     0 program AMIGOinit;
33:S
34:D     1 module CSamigo; {amigo command set}
35:S
36:D     1 import
37:D     1   sysglobals, bkgnd, discHPIB;
38:S
39:D     1 export
40:D     1
41:D     1   type
42:S
43:D     1   amigo_dev_type = {enumerated supported amigo devices}
44:D     1   (HP9895, HP8290X, HP913X_A, HP913X_B, HP913X_C, HP7905, HP7906, HP792D, HP7925);
45:D     1
46:D     1   command_type = {commands supported by the issue_cmd procedure}
47:D     1   ( req_status,      { request status }
48:D     1     req_syndrome,  { request syndrome }
49:D     1     req_log_addr,   { request logical address }
50:D     1     seek_cmd,      { seek }
51:D     1     addr_record_cmd, { address record }
52:D     1     recalibrate_cmd, { recalibrate }
53:D     1     unbuf_read_cmd,  { unbuffered read }
54:D     1     verify_cmd,      { verify }
55:D     1     unbuf_write_cmd, { unbuffered write }
56:D     1     init_d_cmd,      { initialize, setting D bits }
57:D     1     format_cmd,     { format }
58:D     1     buf_read_cmd,    { buffered read }
59:D     1     buf_write_cmd ); { buffered write }
60:D     1
61:D     1   ftc_b_type = {first two command bytes - structure for most commands}
62:D     1   packed record
63:D     1     opcode: byte;
64:D     1     unit: byte;
65:D     1   end;
66:D     1
67:D     1   sl_type = {enumerated status 1 values}
68:D     1   (
69:D     1     normal_completion,
70:D     1     unit_available,
71:D     1     sl_4,
72:D     1     sl_6,
73:D     1     uncorrectable_data_error,
74:D     1     io_program_error,
75:D     1     end_of_cylinder,
76:D     1     overrun,
77:D     1     illegal_access_to_spare_track,
78:D     1     access_not_ready_during_data_operation,
79:D     1     sl_20,
80:D     1     attempt_to_write_on_protected_track,
81:D     1     sl_24,
82:D     1     sl_26,
83:D     1     sl_28,
84:D     1     sl_30,
85:D     1   );

```



```

86:D 1 $page$
87:D 1
88:D 1   tva_type = (three vector address)
89:D 1   packed record
90:D 1     cyl: shortint; { cylinder address }
91:D 1     head: byte; { head address }
92:D 1     sect: byte; { sector address }
93:D 1   end;
94:D 1
95:D 1   status_type = (4 bytes of returned status)
96:D 1   packed record
97:D 1     ( stat 1 - from previous operation )
98:D 1     s: boolean; { 15 spare track bit }
99:D 1     p: boolean; { 14 protected track bit }
100:D 1     d: boolean; { 13 defective track bit }
101:D 1     sl: sl_type; { 12-8 last operation status }
102:D 1     unit: byte; { 7-0 unit number }
103:D 1     ( stat 2 - from specified drive )
104:D 1     star: boolean; { 15 conditions *'ed below }
105:D 1     xx: 0..3; { 14-13 undefined }
106:D 1     tttt: 0..15; { 12-9 disc type bits }
107:D 1     r: boolean; { 8 reserved }
108:D 1     a: boolean; { 7 drive attention }
109:D 1     w: boolean; { 6 write protected }
110:D 1     fmt: boolean; { 5 format switch }
111:D 1     e: boolean; { 4 *drive fault }
112:D 1     f: boolean; { 3 first status bit }
113:D 1     c: boolean; { 2 *seek check }
114:D 1     ss: 0..3; { 1,0 *drive ready status }
115:D 1   end;
116:D 1
117:D 1   syndrome_type = (14 bytes of returned syndrome information)
118:D 1   packed record
119:D 1     sb_pad1: 0..7;
120:D 1     sb_sl: sl_type;
121:D 1     sb_pad2: byte;
122:D 1     sb_tva: tva_type;
123:D 1     sb_offset: shortint;
124:D 1     sb_correction_bytes: packed array[0..5] of char;
125:D 1   end;
126:D 1
127:D 1   map_type = (media addressing parameters)
128:D 1   record
129:D 1     cyl_per_med: shortint; { number of cylinders per medium }
130:D 1     trk_per_cyl: shortint; { number of tracks per cylinder }
131:D 1     sec_per_trk: shortint; { number of sectors per track }
132:D 1   end;
133:D 1
134:D 1   unsigned16 = 0..65535;

```

```

135:D 1 $page$
136:D 1
137:D 1 function device (uep: uep_type): amigo_dev_type;
138:D 1 function MI_controller (uep: uep_type): boolean;
139:D 1 function surface_mode (uep: uep_type): boolean;
140:D 1 procedure get_map (uep: uep_type; var map: map_type);
141:D 1 function records_per_medium (uep: uep_type): integer;
142:D 1 function decoded_addr (uep: uep_type; tva: tva_type): integer;
143:D 1
144:D 1 procedure issue_cmd (uep: uep_type; command: command_type; var cmd_buffer: ftc_b_type);
145:D 1 function dsj (uep: uep_type): byte;
146:D 1 procedure set_file_mask (uep: uep_type);
147:D 1 procedure recalibrate (uep: uep_type);
148:D 1 procedure status (uep: uep_type; var status_bytes: status_type);
149:D 1 procedure syndrome (uep: uep_type; var syndrome_bytes: syndrome_type);
150:D 1 procedure seek (uep: uep_type; record_addr: integer);
151:D 1 procedure addr_record (uep: uep_type; record_addr: integer);
152:D 1 function logical_addr (uep: uep_type): integer;
153:S 1
154:D 1 implement (OSamigo)
155:S 1
156:D 1 var
157:D 1 -4 1 most_recent_status: status_type; {for post-mortem diagnostic purposes only!!!}
158:S 1
159:S 1
160:D 1 function device(uep: uep_type): amigo_dev_type;
161:C 2 begin (device)
162:C 2   case uep^.letter of
163:C 3   'H': device := HP8895;
164:C 3   'N': device := HP8290X;
165:C 3   'U': device := HP913X-A;
166:C 3   'V': device := HP913X-E;
167:C 3   'W': device := HP913X-C;
168:C 3   'Y': device := HP7905;
169:C 3   'C': device := HP7906;
170:C 3   'P': device := HP7920;
171:C 3   'X': device := HP7925;
172:C 3   otherwise device := device(uep, zndevice);
173:C 3   end (case)
174:C 3   end; {device}
175:S 1
176:S 1
177:D 1 function MI_controller(uep: uep_type): boolean;
178:C 2 begin (MI_controller)
179:C 2   MI_controller := device(uep) in [HP7905, HP7906, HP7920, HP7925];
180:C 2   end; {MI_controller}
181:D 1 -4 1
182:S 1
183:D 1 function surface_mode(uep: uep_type): boolean;
184:C 2 begin (surface_mode)
185:C 2   surface_mode := device(uep) in [HP7905, HP7906];
186:C 2   end; {surface_mode}

```



```

187 D -4 1 fpage$
188 S
189 D 1 procedure get_map(uep: uep_type; var map: map_type);
190 D type
191 D device_maps_type = array[HP8290X..HP7925] of map_type;
192 D const
193 D DS9895_map = map_type[ cyl_per_med: 75, trk_per_cyl: 2, sec_per_trk: 30];
194 D SS9895_map = map_type[ cyl_per_med: 73, trk_per_cyl: 1, sec_per_trk: 30];
195 D device_maps = device_maps_type
196 D [(HP8290X) map_type[ cyl_per_med: 33, trk_per_cyl: 2, sec_per_trk: 16],
197 D {HP913X_A} map_type[ cyl_per_med: 152, trk_per_cyl: 4, sec_per_trk: 31],
198 D {HP913X_B} map_type[ cyl_per_med: 305, trk_per_cyl: 4, sec_per_trk: 31],
199 D {HP913X_C} map_type[ cyl_per_med: 305, trk_per_cyl: 6, sec_per_trk: 31],
200 D {HP7905} map_type[ cyl_per_med: 400, trk_per_cyl: 2, sec_per_trk: 48],
201 D {HP7906} map_type[ cyl_per_med: 400, trk_per_cyl: 2, sec_per_trk: 48],
202 D {HP7920} map_type[ cyl_per_med: 800, trk_per_cyl: 5, sec_per_trk: 48],
203 D {HP7925} map_type[ cyl_per_med: 800, trk_per_cyl: 9, sec_per_trk: 64] ];
204 D
205 D -2 var
206 D this_device: amigo_dev_type;
207 D begin (get_map)
208 D this_device := device(uep);
209 D if this_device=HP9895 then (use single/double-sided flag set by status routine)
210 D 3 case uep^dev of
211 D 4 1: map := SS9895_map;
212 D 4 2: map := DS9895_map;
213 D 4 otherwise ioresc_bkgnd(uep, zcatchall);
214 D 4 end (case)
215 D 4 else
216 D 4 map := device_maps[this_device];
217 D 2 end; (get_map)
218 S
219 D 1 function records_per_medium(uep: uep_type): integer;
220 D -6 var
221 D map: map_type;
222 D begin (records_per_medium)
223 D get_map(uep, map);
224 D with map do
225 D 3 records_per_medium := sec_per_trk*trk_per_cyl*cyl_per_med;
226 D 2 end; (records_per_medium)
227 S
228 D 1 function dsj(uep: uep_type): byte;
229 D -2 var
230 D dsj_byte: packed record b: byte; end;
231 D -2 const
232 D dsj_sec = 16;
233 D -2 begin (dsj)
234 D HP18short_msge_in(uep, dsj_sec, addr(dsj_byte), sizeof(dsj_byte));
235 D dsj := dsj_byte.b;
236 D 2 end; (dsj)
237 C

```

```

238 D -4 1 $page$
239 D -4
240 D 1 procedure issue_cmd(uep: uep_type; command: command_type; var cmd_buffer: ftc_b_type);
241 D type
242 D ctet = (command table entry type)
243 D packed record
244 D sec: shortint; { secondary command }
245 D oc: byte; { opcode }
246 D nb: byte; { number of data bytes }
247 D end;
248 D command_table_type = packed array[command_type] of ctet;
249 D const
250 D command_table = command_table_type
251 D [ {req_status} ctet[sec: 8, oc: 03, nb: 2],
252 D {req_syndrome} ctet[sec: 8, oc: 13, nb: 2],
253 D {req_log_addr} ctet[sec: 8, oc: 20, nb: 2],
254 D {seek_cmd} ctet[sec: 8, oc: 02, nb: 6],
255 D {addr_record_cmd} ctet[sec: 8, oc: 12, nb: 6],
256 D {recalibrate_cmd} ctet[sec: 8, oc: 01, nb: 2],
257 D {unbuf_read_cmd} ctet[sec: 8, oc: 05, nb: 2],
258 D {verify_cmd} ctet[sec: 8, oc: 07, nb: 4],
259 D {unbuf_write_cmd} ctet[sec: 8, oc: 08, nb: 2],
260 D {init_d_cmd} ctet[sec: 8, oc: 43, nb: 2],
261 D {format_cmd} ctet[sec: 12, oc: 24, nb: 5],
262 D {buf_read_cmd} ctet[sec: 10, oc: 05, nb: 2],
263 D {buf_write_cmd} ctet[sec: 9, oc: 08, nb: 2] ];
264 D begin (issue_cmd)
265 D with cmd_buffer, command_table[command] do
266 D 3 begin
267 D 3 opcode := oc;
268 D 3 unit := uep^du;
269 D 3 HP18short_msge_out(uep, sec, addr(cmd_buffer), nb);
270 D 3 end; (with)
271 D 2 end; (issue_cmd)
272 S
273 D 1 procedure set_file_mask(uep: uep_type);
274 D type
275 D sfm_cmd_type = (set file mask command)
276 D packed record
277 D oc: byte;
278 D mask: byte;
279 D end;
280 D sfm_cmd_array_type = array[boolean] of sfm_cmd_type;
281 D const
282 D sfm_sec = 8; {secondary}
283 D sfm_oc = 18; {op code}
284 D sfm_cmd_array = sfm_cmd_array_type
285 D [ {false: cylinder mode} sfm_cmd_type[ oc: sfm_oc, mask: 7 ],
286 D {true: surface mode} sfm_cmd_type[ oc: sfm_oc, mask: 5 ] ];
287 D var
288 D sfm_cmd: sfm_cmd_type;
289 D begin (set_file_mask)
290 D sfm_cmd := sfm_cmd_array[surface_mode(uep)];
291 D HP18short_msge_out(uep, sfm_sec, addr(sfm_cmd), sizeof(sfm_cmd));
292 D end; (set_file_mask)
293 C

```



```

294:D -4 1 $page$
295:D -4 1
296:D 1 procedure recalibrate(uep: uep_type);
297:D 2 var
298:D 2     recalibrate_cmd_buf: ftc_b_type;
299:C -2 2     begin {recalibrate}
300:C 2         issue_cmd(uep, recalibrate_cmd, recalibrate_cmd_buf);
301:C 2     end; {recalibrate}
302:D -4 1
303:D 1
304:D 1 procedure status(uep: uep_type; var status_bytes: status_type);
305:D 2 var
306:D 2     status_cmd_buf: ftc_b_type;
307:D -2 2     const
308:D -2 2         send_sts_sec = 8;
309:C -2 2     begin {status}
310:C 2         issue_cmd(uep, req_status, status_cmd_buf);
311:C 2         if not MI_controller(uep) then HPIWait_for_ppol(uep);
312:C 2         HPIshort_msge_in(uep, send_sts_sec, addr(status_bytes), sizeof(status_bytes));
313:C 2         most_recent_status := status_bytes; {for post-mortem diagnostic purposes only!}
314:C 2         with uep^ do
315:C 3             case device(uep) of
316:C 4                 HP8895: (use the otherwise undefined devid field to indicate...)
317:C 4                     if status_bytes.tttt in [5,6]
318:C 5                         then devid := 2; {double-sided disc}
319:C 5                         else devid := 1; {single-sided disc}
320:C 4                 HP8290X: (use the otherwise undefined devid field to indicate...)
321:C 4                     devid := ord(status_bytes.r); {Sparrow (1) versus Chinook (0)}
322:C 4                 otherwise
323:C 4                     {do nothing};
324:C 4             end; {case}
325:C 2         end; {status}
326:D -4 1
327:D 1
328:D 1 procedure syndrome(uep: uep_type; var syndrome_bytes: syndrome_type);
329:D 2 var
330:D 2     syndrome_cmd_buf: ftc_b_type;
331:D -2 2     const
332:D -2 2         send_syn_sec = 8;
333:C -2 2     begin {syndrome}
334:C 2         issue_cmd(uep, req_syndrome, syndrome_cmd_buf);
335:C 2         HPIshort_msge_in(uep, send_syn_sec, addr(syndrome_bytes), sizeof(syndrome_bytes));
336:C 2     end; {syndrome}

```

```

337:D -4 1 $page$
338:D -4 1
339:D 1 function coded_addr(uep: uep_type; record_addr: integer): tva_type;
340:D 2 var
341:D 2     map: map_type;
342:D -6 2     track: integer;
343:C -10 2     begin {coded_addr}
344:C 2         get_map(uep, map);
345:C 2         with map do
346:C 3             begin
347:C 3                 coded_addr.sect := record_addr mod sec_per_trk;
348:C 3                 track := record_addr div sec_per_trk;
349:C 3                 if surface_mode(uep) then
350:C 4                     begin {select proper 7905/06 logical "volume"}
351:C 4                         coded_addr.head := track div cyl_per_med + 2*uep^.dv;
352:C 4                         coded_addr.cyl := track mod cyl_per_med;
353:C 4                     end {then}
354:C 4                 else
355:C 4                     begin
356:C 4                         coded_addr.head := track mod trk_per_cyl;
357:C 4                         coded_addr.cyl := track div trk_per_cyl;
358:C 4                     end; {else}
359:C 3             end; {with}
360:C 2         end; {coded_addr}
361:D 1
362:D 1
363:D 1 function decoded_addr(uep: uep_type; tva: tva_type): integer;
364:D 2 var
365:D 2     map: map_type;
366:D -6 2     track: integer;
367:C -10 2     begin {decoded_addr}
368:C 2         get_map(uep, map);
369:C 2         with tva, map do
370:C 3             begin
371:C 3                 if surface_mode(uep)
372:C 3                     then track := (head-2*uep^.dv)*cyl_per_med+cyl
373:C 4                     else track := cyl*trk_per_cyl+head;
374:C 3                 decoded_addr := track*sec_per_trk+sect;
375:C 3             end; {with}
376:C 2         end; {decoded_addr}

```



```

377:D -4 1 $page$
378:S
379:D 1 procedure seek(uep: uep_type; record_addr: integer);
380:D var
381:D seek_cmd_buf:
382:D packed_record
383:D ftc_b: ftc_b_type;
384:D tva: tva_type;
385:D -6 end;
386:D begin (seek)
387:C seek_cmd_buf.tva := coded_addr(uep, record_addr);
388:C issue_cmd(uep, seek_cmd, seek_cmd_buf.ftc_b);
389:C end; (seek)
390:S
391:S
392:D 1 procedure addr_record(uep: uep_type; record_addr: integer);
393:D var
394:D addr_record_cmd_buf:
395:D packed_record
396:D ftc_b: ftc_b_type;
397:D tva: tva_type;
398:D -6 end;
399:C begin (addr_record)
400:C addr_record_cmd_buf.tva := coded_addr(uep, record_addr);
401:C issue_cmd(uep, addr_record_cmd, addr_record_cmd_buf.ftc_b);
402:C end; (addr_record)
403:S
404:S
405:D 1 function logical_addr(uep: uep_type): integer;
406:D var
407:D ladd_cmd_buf ftc_b_type;
408:D tva: tva_type;
409:D -6 const
410:D send_addr_sec = 8;
411:C begin (logical_addr)
412:C issue_cmd(uep, req_log_addr, ladd_cmd_buf);
413:C if not HI_controller(uep) then HPIBwait_for_ppol(uep);
414:C HPIBshort_msg_in(uep, send_addr_sec, addr(tva), sizeof(tva));
415:C logical_addr := decoded_addr(uep, tva);
416:C end; (logical_addr)
417:S
418:S
419:C 1 end; (CSamigo)

```

```

420:D 1 $page$
421:S
422:D 1 module amigodvr;
423:S
424:D 1 import
425:D 1 sysglobals, drvasm, bkgnd, discHPIB, CSamigo;
426:S
427:D 1 export
428:D 1 procedure get_letter(uep: uep_type; ident: shortint; var letter: char);
429:D 1
430:D 2 procedure amigoio(fp: fibp; request: amrequesttype;
431:D anyvar buffer: window; length, position: integer);
432:S
433:S 1 implement (amigodvr)
434:S
435:S
436:S
437:S 1 procedure used by CTABLE for self-configuring
438:D 1
439:D 1 procedure get_letter(uep: uep_type; ident: shortint; var letter: char);
440:D const
441:D ident_table_entries = 7;
442:D var
443:D -2 index: shortint;
444:D -6 status_bytes: status_type;
445:D -6 type
446:D -6 device_table_type = array[0..3] of char;
447:D -6 itet = (ident_table_entry_type)
448:D -6 record
449:D -6 ident: shortint;
450:D -6 letter: char;
451:D -6 end;
452:D -6 ident_table_type = array[1..ident_table_entries] of itet;
453:D -6 const
454:D -6 device_table = device_table_type
455:D -6 [(0) 'C', (1) 'P', (2) 'V', (3) 'X' ];
456:D -6 ident_table = ident_table_type
457:D -6 [(HP8895) itet[ ident: 0*256+129 ($0081), letter: 'H' ],
458:D -6 (HP8290X) itet[ ident: 1*256+ 4 ($0104), letter: 'N' ],
459:D -6 (HP813XA) itet[ ident: 1*256+ 6 ($0106), letter: 'U' ],
460:D -6 (HP813XB) itet[ ident: 1*256+ 10 ($010A), letter: 'V' ],
461:D -6 (HP813XC) itet[ ident: 1*256+ 15 ($010F), letter: 'W' ],
462:D -6 (MAC) itet[ ident: 0*256+ 2 ($0002), letter: 'X' ],
463:D -6 (IDC) itet[ ident: 0*256+ 3 ($0003), letter: 'X' ] ];
464:D -6
465:D 2 begin (get_letter)
466:D 2 letter := chr(255); (initially undefined)
467:D 2 for index := 1 to ident_table_entries do
468:D 3 if ident=ident_table[index].ident then
469:D 4 letter := ident_table[index].letter;
470:D 2 if letter=chr(255) then ioresc_bkgnd(uep, znodevice);
471:D 2 uep.letter := letter; (for determining ppol wait in status routine)
472:D 2 if dsj(uep)<>0 then (don't worry about it);
473:D 2 HPIBamigo_clear(uep);
474:D 2 HPIBwait_for_ppol(uep);
475:D 2 status(uep, status_bytes);
476:D 2 if dsj(uep)<>0 then ioresc_bkgnd(uep, zcatchall);
477:D 2 if status_bytes.s=2 then ioresc_bkgnd(uep, znodevice); ("unit not present or power off")
478:S

```



```

480:C      2      if letter='X' then (determine which 7906 family member it really is)
481:C      3      begin
482:C      4      if not (status_bytes.tttt in [0..3]) then
483:C      5      ioresc_bkgnd(uep, znodevice); {unrecognized unit type}
484:C      3      letter := device_table[status_bytes.tttt];
485:C      3      end; {if}
486:S
487:C      2      end; {get_letter}
488:S
489:S
490:D      1      procedure clear_unit(uep:uep_type ;
491:D      2      type
492:D      3      device_ident_type = array[HP9895..HP913X_C] of shortint;
493:D      3      device_table_type = array[0..3] of amigo_dev_type;
494:D      3      const
495:D      4      device_ident = device_ident_type
496:D      5      [ (HP9895) 0*256+129, ($0031)
497:D      6      (HP8290X) 1*256+ 4, ($0124)
498:D      7      (HP913X_A) 1*256+ 6, ($0126)
499:D      8      (HP913X_B) 1*256+10, ($012A)
500:D      9      (HP913X_C) 1*256+15, ($013F) ];
501:D      3      MAC_ident = 0*256+2; ($0002)
502:D      3      IDC_ident = 0*256+3; ($0003)
503:D      3      device_table = device_table_type
504:D      4      [ (0) HP7906, (1) HP7320, (2) HP7905, (3) HP7925 ];
505:D      3      var
506:D      4      dev: amigo_dev_type;
507:D      4      ident: shortint;
508:D      4      dummy_dsj: byte;
509:D      4      status_bytes: status_type;
510:C      3      begin {clear_unit}
511:C      4      dev := device(uep);
512:C      4      ident := HPI8amigo_identify(uep);
513:C      4      if MI_controller(uep) then {check for MAC or IDC controller}
514:C      5      begin
515:C      6      if not ((ident=MAC_ident) or ((ident=IDC_ident) and Simon_DMA(uep))) then
516:C      7      ioresc_bkgnd(uep, znodevice);
517:C      8      end {then}
518:C      5      else {require EXACT device/ident match}
519:C      6      if ident<>device_ident[dev] then ioresc_bkgnd(uep, znodevice);
520:C      6      if dev=HP8290X then {avoid the amigo clear; it takes too much time!}
521:C      7      dummy_dsj := dsj(uep) {just remove the power-on holdoff}
522:C      6      else {go ahead and do the hard clear}
523:C      7      begin
524:C      8      HPI8amigo_clear(uep);
525:C      8      HPI8wait_for_ppol(uep);
526:C      8      end; {else}
527:C      5      status_bytes := status_bytes;
528:C      5      if dsj(uep)<0 then ioresc_bkgnd(uep, zcatchall);
529:C      5      if status_bytes.ss=2 then ioresc_bkgnd(uep, znodevice); {"unit not present or power off"}
530:C      5      if MI_controller(uep) then {we need to check the exact type of THIS particular unit}
531:C      6      begin
532:C      7      if not (status_bytes.tttt in [0..3]) then
533:C      8      ioresc_bkgnd(uep, znodevice); {unrecognized unit type}
534:C      8      if dev<>device_table[status_bytes.tttt] then
535:C      9      ioresc_bkgnd(uep, znodevice); {wrong unit type}
536:C      7      end; {then}
537:C      5      end; {clear_unit}

```

```

538:D      1      $page$
539:S
540:S
541:S      {
542:D      1      procedures in the background transfer chain
543:D      2      1
544:D      2      1 procedure initial_seek      (uep: uep_type); forward;
545:D      2      1 procedure enter_transfer_chain (uep: anyptr); forward;
546:D      2      1 procedure issue_transfer_request (uep: uep_type); forward;
547:D      2      1 procedure initiate_data_transfer (uep: anyptr); forward;
548:D      2      1 procedure upon_data_transfer_comp(uep: anyptr); forward;
549:D      2      1 procedure check_dsj      (uep: anyptr); forward;
550:S
551:S
552:S      {
553:D      1      main driver procedure
554:D      2      1
555:D      2      1 procedure amigoio;
556:D      2      2
557:D      2      3      var
558:D      2      4      uep: uep_type;
559:D      2      4      ident: shortint;
560:D      2      4      asynchronous: boolean;
561:C      2      3      begin {amigoio}
562:C      3      uep := addr(unitable^[fp^.funit]);
563:C      3      asynchronous := (request=startread) or (request=startwrite);
564:C      3      if uep^.offline then
565:C      4      ioresult := ord(znodevice)
566:C      4      else
567:C      5      try
568:C      6      ioresult := ord(inoerror);
569:C      6      case request of
570:C      7      clearunit:
571:C      8      begin
572:C      9      uep^.umediavalid := false;
573:C      9      unit_wait(uep);
574:C      9      ioresult := ord(inoerror); {forget any previous error}
575:C      9      allocate_bkgnd_info(uep);
576:C      9      HPI8check_sc(uep);
577:C      9      clear_unit(uep);
578:C      9      deallocate_bkgnd_info(uep);
579:C      9      end;
580:C      7      unitstatus:
581:C      8      fp^.fbusy := unit_busy(uep);
582:C      7      flush:
583:C      8      {do nothing};
584:C      7      readbytes, writebytes, startread, startwrite:
585:C      8      begin {transfer operations}
586:C      9      unit_wait(uep);
587:C      9      ioresult := ord(inoerror); {forget any previous error}
588:C      9      allocate_bkgnd_info(uep);
589:C      9      end;

```



```

594:C      5      with bip_type(uep^.dvrtemp)^ do
595:C      6      begin
596:C      6          feot := fp^.feot;      (end of transfer procedure)
597:C      6          fibptr := fp;      (parameter to the eot procedure)
598:C      6          async := asynchronous;      (determines whether or not to call eot proc)
599:C      6
600:C      6      if Simon_no_dma(uep) then
601:C      7          ioresc_bkgnd(uep, zbaddma);
602:C      7      if uep^.ureportchange and not uep^.umediavalid then
603:C      7          ioresc_bkgnd(uep, zmediumchanged);
604:C      7      if position mod 256 < 0 then
605:C      7          ioresc_bkgnd(uep, zbaddmode);
606:C      6      if (position < 0) or (length < 0) or (position+length > fp^.fpeof) then
607:C      7          ioresc_bkgnd(uep, ieof);
608:C      6
609:C      6      HPIBcheck_sc(uep);
610:C      6      ident := HPIBamigo_identify(uep);      (confirm device present)
611:C      6      if dsj(uep) < 0 then (do nothing);      (remove power-on holdoff if any)
612:C      6
613:C      6      if length=0 then
614:C      7          deallocate_bkgnd_info(uep)      (nothing to transfer)
615:C      7      else
616:C      7          begin
617:C      7              read_operation := (request=readbytes) or (request=startread);
618:C      7              xfr_chain_semaphore := false;
619:C      7              bx_tries := 0;
620:C      7              bx_strt_rcrd := (position+fp^.fileid+uep^.byteoffset) div 256;
621:C      7              bx_bufptr := addr(buffer);
622:C      7              bx_length := length;
623:C      7              initial_seek(uep);      (initiate the transfer)
624:C      7          end;      (else)
625:C      6      end;      (with)
626:C      6
627:C      5      if not asynchronous then
628:C      6      begin
629:C      6          unit_wait(uep);
630:C      6          uep^.dvrtemp := ord(inoerror);      (report synchronous errors only once)
631:C      6          end;      (if)
632:C      5      end;      (transfer operations)
633:C      5
634:C      5      otherwise      (unrecognized request)
635:C      5          ioresult := ord(ibadrequest);
636:C      5
637:C      5      end;      (cases)
638:C      4
639:C      4      recover
640:C      4      begin
641:C      4          abort_bkgnd_process(uep);
642:C      4          ioresult := uep^.dvrtemp;
643:C      4          if not asynchronous then
644:C      5              uep^.dvrtemp := ord(inoerror);      (report synchronous errors only once)
645:C      4          end;      (recover)
646:C      2
647:C      2      end;      (amigoio)
648:C      2

```

```

652:O      1  $page$
653:S
654:D      1  procedure initial_seek(uep: uep_type);
655:O      2  var
656:D      2  status_bytes: status_type;
657:O      2  -4      begin      (initial_seek)
658:O      2  begin
659:O      2  if device(uep)=HP9895 then      (read status to determine single or double-sided)
660:C      3  begin
661:C      3  status(uep, status_bytes);
662:C      3  with status_bytes do      (specifically disallow non HP-formatted discs)
663:C      4  begin
664:C      4  if f then
665:C      5  begin
666:C      5  uep^.umediavalid := false;
667:C      5  if uep^.ureportchange then
668:C      6  ioresc_bkgnd(uep, zmediumchanged);
669:C      5  end;      (if)
670:C      4  if (ss=0) and not (tttt in [2,6]) then
671:C      5  ioresc_bkgnd(uep, zuninitialized);
672:C      4  end;      (with)
673:C      3  end;      (if)
674:C      2  end;
675:C      2  with bip_type(uep^.dvrtemp)^ do
676:C      3  begin
677:C      3  if MI_controller(uep) then
678:C      4  begin
679:C      4  set_file_mask(uep);
680:C      4  buffered_transfer := not Simon_DMA(uep);
681:C      4  HPIBwait_for_ppol(uep);      (shouldn't take very long!)
682:C      4  end      (then)
683:C      4  else
684:C      4  if device(uep)=HP8290X
685:C      5  then buffered_transfer := uep^.devid=0      (Sparrow (1) versus Chinook (0))
686:C      5  else buffered_transfer := false;
687:C      4  end
688:C      4  if device(uep) in [HP913X_A..HP913X_C]
689:C      5  then addr_record(uep, bx_strt_rcrd);      (allows greater overlapping with 9914)
690:C      4  else seek
691:C      5  (uep, bx_strt_rcrd);
692:C      4  end
693:C      3  HPIBupon_ppol_resp(uep, enter_transfer_chain);
694:C      3  end;      (with)
695:C      2  end;      (initial_seek)
696:C      2
697:C      2
698:S
699:S
700:D      1  procedure enter_transfer_chain(uep: anyptr);
701:C      2  begin      (enter_transfer_chain)
702:C      2  with bip_type(uep^.dvrtemp)^ do
703:C      3  if not test_and_toggle(xfr_chain_semaphore) then
704:C      4  repeat
705:C      5  issue_transfer_request(uep);
706:C      5  until test_and_toggle(xfr_chain_semaphore);
707:C      3  end;      (enter_transfer_chain)

```



```

708:D 1 $page$
709:S
710:D 1 procedure issue_transfer_request(uep: uep_type);
711:D 2 const
712:D 2   sect_per_surf = 400*48; {only valid for 7805/06!!!}
713:D 2 var
714:D -2 2   transfer_command: command_type;
715:D -4 2   transfer_cmd_buf: ftc_b_type;
716:D -8 2   max_tfr_length: integer;
717:D -12 2   remaining_surf_bytes: integer;
718:D -13 2   wait_for_ppol: boolean;
719:C 2   begin {issue_transfer_request}
720:C 2   with bip_type(uep^.dvrtemp)^ do
721:C 3   try
722:C 4   if buffered_transfer then
723:C 4   begin
724:C 5   if read_operation
725:C 5   then transfer_command := buf_read_cmd
726:C 5   else transfer_command := buf_write_cmd;
727:C 5   max_tfr_length := 256;
728:C 5   end {Then}
729:C 5   else
730:C 5   begin
731:C 5   if read_operation
732:C 5   then transfer_command := unbuf_read_cmd
733:C 5   else transfer_command := unbuf_write_cmd;
734:C 5   if HI_controller(uep) then
735:C 5   begin
736:C 5   max_tfr_length := 65536; {max DMA burst length}
737:C 5   if surface_mode(uep) then {don't try to cross a surface boundary}
738:C 5   begin
739:C 5   remaining_surf_bytes := (sect_per_surf-bx_stirt.rcrd mod sect_per_surf)*256;
740:C 5   if remaining_surf_bytes < max_tfr_length then
741:C 5   max_tfr_length := remaining_surf_bytes;
742:C 5   end; {then}
743:C 5   end {then}
744:C 5   else
745:C 5   max_tfr_length := maxint;
746:C 5   end; {else}
747:C 5   end
748:C 5   if bx_length <= max_tfr_length
749:C 5   then bx_tfr_length := bx_length
750:C 5   else bx_tfr_length := max_tfr_length;
751:C 5   end
752:C 5   wait_for_ppol := buffered_transfer or (device(uep)=HP8290X);
753:C 5   issue_cmd(uep, transfer_command, transfer_cmd_buf);
754:C 5   if wait_for_ppol {computed above because of critical MAC/IDC timing;}
755:C 5   then HPiBupon_ppol_resp(uep, initiate_data_transfer)
756:C 5   else initiate_data_transfer(uep);
757:C 5   recover
758:C 5   abort_bkgnd_process(uep);
759:C 2   end; {issue_transfer_request}

```

```

760:D 1 $page$
761:S
762:D 1 procedure initiate_data_transfer(uep: anyptr);
763:D 2 const
764:D 2   tfr_data_sec = 0;
765:C 2   begin {initiate_data_transfer}
766:C 2   with bip_type(uep_type(uep)^.dvrtemp)^ do
767:C 3   try
768:C 4   HPiBupon_dxfr_comp(uep, tfr_data_sec, bx_bufptr, bx_tfr_length, upon_data_transfer_comp);
769:C 4   recover
770:C 4   abort_bkgnd_process(uep);
771:C 2   end; {initiate_data_transfer}
772:S
773:S
774:D 1 procedure upon_data_transfer_comp(uep: anyptr);
775:C 2   begin {upon_data_transfer_comp}
776:C 2   try
777:C 3   if bip_type(uep_type(uep)^.dvrtemp)^.iores <> ioinerror then escape(-10);
778:C 3   HPiBupon_ppol_resp(uep, check_ds);
779:C 3   recover
780:C 3   abort_bkgnd_process(uep);
781:C 2   end; {upon_data_transfer_comp}
782:S
783:D 1
784:D 1 procedure check_ds(uep: anyptr);
785:D 2 var
786:D 2   transfer_successful: boolean;
787:D -1 2   const
788:D -1 2   maxtries = 10;
789:D -1 2
790:D 2 procedure process_errors(uep: uep_type);
791:D 3 var
792:D -4 3   status_bytes: status_type;
793:D -18 3   syndrome_bytes: syndrome_type;
794:D -24 3   cb_ptr: charptr;
795:D -24 3   cb_index: shortint;
796:D -28 3   e_rcrd: integer;
797:D -32 3   possible_bytes_transferred: integer;
798:C 3   begin {process_errors}
799:C 3   with bip_type(uep^.dvrtemp)^ do
800:C 4   begin
801:C 4   status(uep, status_bytes);
802:C 4   if ds(uep) < 0 then ioresc_bkgnd(uep, zcatchall);
803:C 4   with status_bytes do
804:C 5   case s1 of
805:C 6   (retryable_errors)
806:C 6   cylinder_compare_error,
807:C 6   uncorrectable_data_error,
808:C 6   head_sector_compare_error,
809:C 6   end_of_cylinder,
810:C 6   sync_bit_not_received_in_time,
811:C 6   overrun,
812:C 6   possibly_correctable_data_error,
813:C 6   illegal_access_to_spare_track,
814:C 6   defective_track,
815:C 6   access_not_ready_during_data_operation:
816:C 6   begin {retryable_errors case}
817:C 6   if si < overrun
818:C 7   then e_rcrd := bx_stirt.rcrd {addr untrustworthy after overrun}
819:C 7   else e_rcrd := logical_addr(uep);

```



```

82):C      t      if si=cylinder_compare_error then
83):C      f      if MI_controller(uep) then (recalibrate & retry)
84):C      s      begin
85):C      s      recalibrate(uep);
86):C      s      HPIWait_for_ppol(uep); (shouldn't happen very often!)
87):C      s      end (then)
88):C      s      else if e_rcrd=bx_stirt_rcrd then (don't retry...)
89):C      f      ioresc_bkgnd(uep, znoblock); (Chinook takes 25 secs/retry!!!)
90):C      t      if e_rcrd=bx_stirt_rcrd then (careful with MAC/IOC verify address!)
91):C      ?      begin
92):C      7      bx_tries := bx_tries+1;
93):C      7      transfer_successful := false; (unless correctable below)
94):C      7      if (si=possibly_correctable_data_error) and (bx_tries>5) then
95):C      8      with syndrome_bytes do
96):C      9      begin
97):C      9      syndrome(uep, syndrome_bytes);
98):C      10     if (sb_si=possibly_correctable_data_error) and
99):C      10     (decoded_addr(uep, sb_tva)=e_rcrd) and
100):C     10     (sb_offset=0) and
101):C     10     (sb_offset<=125) then (it's correctable!)
102):C     10     begin
103):C     10     cb_ptr := addr(bx_bufptr^, 2*sb_offset);
104):C     10     cb_index := 0;
105):C     10     while (cb_index<6) and
106):C     11     (integer(cb_ptr)<integer(bx_bufptr)+bx_tfr_length) do
107):C     11     begin
108):C     11     eor(sb_correction_bytes[cb_index], cb_ptr);
109):C     11     cb_ptr := addr(cb_ptr^, 1);
110):C     11     cb_index := cb_index+1;
111):C     11     end; (while)
112):C     10     e_rcrd := e_rcrd+1; (this record has been corrected!)
113):C     10     bx_tries := 0; (no attempts made on the next record yet)
114):C     10     transfer_successful := true; (at least partially!)
115):C     10     end; (then)
116):C     9     end; (with)
117):C     7     end (then); (with)
118):C     7     else
119):C     7     begin
120):C     7     bx_tries := 1; (first attempt on this record)
121):C     7     transfer_successful := true; (at least partially!)
122):C     7     end; (else)
123):C     6     if transfer_successful then
124):C     7     begin
125):C     7     possible_bytes_transferred := (e_rcrd-bx_stirt_rcrd)*256;
126):C     7     if bx_tfr_length>possible_bytes_transferred then
127):C     8     bx_tfr_length := possible_bytes_transferred;
128):C     7     end (then)
129):C     7     else
130):C     7     if bx_tries>=maxtries then
131):C     6     if si in [uncorrectable_data_error, possibly_correctable_data_error]
132):C     6     then ioresc_bkgnd(uep, zbadblock)
133):C     6     else ioresc_bkgnd(uep, znoblock);
134):C     6     end; (retryable errors case)
135):C     6     {immediate escape errors}
136):C     6     illegal_drive_type,
137):C     6     unit_unavailable.

```

```

880):C      6      ioresc_bkgnd(uep, znodevice);
881):C      6      attempt_to_write_on_protected_track;
882):C      6      ioresc_bkgnd(uep, zprotected);
883):C      6      (errors requiring status 2 processing)
884):C      6      status_2_error:
885):C      6      begin
886):C      6      if f then
887):C      7      uep^.umediavalid := false;
888):C      6      if e then ioresc_bkgnd(uep, zbadhardware);
889):C      6      case ss of
890):C      7      1: ioresc_bkgnd(uep, znotready);
891):C      7      2: ioresc_bkgnd(uep, znodevice);
892):C      7      3: ioresc_bkgnd(uep, znomedium);
893):C      7      otherwise (test further conditions below);
894):C      7      end; (case)
895):C      6      if not read_operation and w then
896):C      7      ioresc_bkgnd(uep, zprotected);
897):C      6      if not MI_controller(uep) and not (tttt in [2,6]) then
898):C      7      ioresc_bkgnd(uep, zuninitialized);
899):C      6      if f then
900):C      7      begin
901):C      7      if uep^.ureportchange then
902):C      8      ioresc_bkgnd(uep, zmediumchanged);
903):C      7      bx_tries := bx_tries+1;
904):C      7      if bx_tries>1 then
905):C      8      ioresc_bkgnd(uep, zcatchall);
906):C      7      transfer_successful := false;
907):C      7      end (then)
908):C      7      else
909):C      7      ioresc_bkgnd(uep, zcatchall);
910):C      7      end;
911):C      6      drive_attention:
912):C      6      begin
913):C      6      if e then ioresc_bkgnd(uep, zbadhardware);
914):C      6      if c then
915):C      7      begin
916):C      7      e_rcrd := logical_addr(uep);
917):C      7      if e_rcrd>(bx_stirt_rcrd+(bx_tfr_length-1)div 256) then
918):C      8      transfer_successful := true (already transferred enough bytes)
919):C      7      else
920):C      8      if e_rcrd>=records_per_medium(uep)
921):C      8      then ioresc_bkgnd(uep, znosuchblk)
922):C      8      else ioresc_bkgnd(uep, znoblock);
923):C      7      end (then)
924):C      7      else
925):C      7      ioresc_bkgnd(uep, zcatchall);
926):C      7      end;
927):C      6      (other errors)
928):C      6      otherwise
929):C      6      ioresc_bkgnd(uep, zcatchall);
930):C      6      end; (case)
931):C      4      end; (with)
932):C      3      end; (process_errors)

```



```

935:D -1 2 $page$
936:D -1 2
937:C 2 begin (check_dsj)
938:C 2 with bip_type(uep_type(uep)^.dvrtmp)^ do
939:C 3 try
940:C 4
941:C 4 if dsj(uep)=0 then
942:C 5 if bdx_pre_eoi
943:C 6 then ioresc_bkgnd(uep, zcatchall) {unresolved premature eoi}
944:C 6 else transfer_successful := true
945:C 6 else
946:C 5 process_errors(uep); {will set/clear transfer_successful, or escape}
947:C 4
948:C 4 if transfer_successful then
949:C 5 begin
950:C 5 bx_stirt_rcrd := bx_stirt_rcrd+bx_tfr_length div 256;
951:C 5 bx_bufptr := addr(bx_bufptr^, bx_tfr_length);
952:C 5 bx_length := bx_length-bx_tfr_length;
953:C 5 end; {then}
954:C 4
955:C 4 if bx_length>0 then
956:C 5 if Hl_controller(uep) or not transfer_successful then
957:C 6 begin
958:C 6 if device(uep) in [HP913X.A..HP913X.C, HP7905..HP7925]
959:C 7 then addr_record(uep, bx_stirt_rcrd)
960:C 7 else seek (uep, bx_stirt_rcrd);
961:C 6 HPiBupon_ppol_resp(uep, enter_transfer_chain);
962:C 6 end {then}
963:C 6 else
964:C 6 enter_transfer_chain(uep)
965:C 6 else
966:C 5 deallocate_bkgnd_info(uep);
967:C 4
968:C 4 recover
969:C 4 abort_bkgnd_process(uep);
970:C 2 end; {check_dsj}
971:C
972:C
973:C 1 end; {amigodvr}
974:S
975:S
976:S
977:D 1 { program AMIGOinit }
978:D
979:D 1 import
980:D 1 loader;
981:S
982:C 1 begin {AMIGOinit}
983:C 1 markuser;
984:C 1 end. {AMIGOinit}
985:S
986:C 1

```

No errors. No warnings.

***** Nonstandard language features enabled *****

ASCII

Description

ASCII contains the access method (AM) used to read and write text to a file of type ASC. ASCII also contains the routine necessary to install itself in the system.

Usage

ASCII is used by the file system to read and write .ASC files.

Requirements

SYSGLOBALS and ASM.

Notes

ASCII can do “direct access” reads on record boundaries. This is required by the Compiler to allow INCLUDE files.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

10:D 0 $SYSPROG$ $DEBUG OFF, RANGE OFF, OVFLCHECK OFF, STACKCHECK OFF$
11:D 0 program installasci1;
12:D 1 module asci1module;
13:D 1 import sysglobals, asm;
14:D 1 export
15:D 1   procedure asci1am(fp : fbp; request: amrequesttype;
16:D 1   anyvar buffer:window; bufsize,position:integer);
17:D 1 implement
18:D 1 const
19:D 1   buflengeth = fbksize;
20:D 1   sectorsize = 256;
21:S 1
22:D 1 procedure asci1am(fp: fbp; request: amrequesttype;
23:D 1   anyvar buffer: window; bufsize,position: integer);
24:D 1 var
25:D 1   bufend : integer;
26:S -4
27:D 2 procedure bufparams;
28:D 2 begin
29:D 2   with fp^ do
30:D 2   if (flastpos+buflengeth)>fpeof then bufend := fpeof
31:D 2   else bufend := flastpos + buflengeth;
32:D 2 end;
33:S 2
34:D 2 procedure readfile;
35:D 2 begin
36:D 2   if request<>readtoeof then ioresult := ord(ieof);
37:D 2   fp^.fpos := fp^.fleaf; { fix fpos }
38:D 2   escape(0);
39:D 2 end;
40:S 2
41:D 2 procedure flushbuffer;
42:D 2 begin
43:D 2   ioresult := 0;
44:D 2   with fp^, unitable^[funit] do
45:D 2   begin { write out the buffer }
46:D 2     call(tm,fp,WRITEBYTES,fbuffer,bufend-flastpos,flastpos);
47:D 2     if ioresult<>0 then escape(0);
48:D 2     fbufchanged := false;
49:D 2   end;
50:D 2 end;
51:S 2
52:D 2 procedure loadbuffer(posit: integer);
53:D 2 begin
54:D 2   ioresult:=0;
55:D 2   with fp^, unitable^[funit] do
56:D 2   begin
57:D 2     if fbufchanged then flushbuffer;
58:D 2     flastpos := (posit div sectorsize) * sectorsize; bufparams;
59:D 2     call(tm,fp,READBYTES,fbuffer,bufend-flastpos,flastpos);
60:D 2     if ioresult<>0 then escape(0);
61:D 2   end;
62:D 2 end;
63:S 2
64:D 2 procedure seekposit(posit: integer);
65:D 2 begin
66:D 2   with fp^ do
67:D 2   begin
68:D 2     if (posit<flastpos) or (posit>=bufend) then loadbuffer(posit);
69:D 2     fpos:=posit;

```

```

61:C 4 end;
62:C 3 end;
63:S 3
64:D 2 procedure wendbuffer;
65:D 2 begin { append access only }
66:D 2   with fp^ do
67:D 2   begin
68:D 2     { write all but last sector }
69:D 2     fbufchanged := true;
70:D 2     if (bufend-flastpos)>sectorsize then
71:D 2     begin
72:D 2       bufend := bufend - sectorsize;
73:D 2       flushbuffer;
74:D 2       moveleft(fbuffer[bufend-flastpos],fbuffer[0],sectorsize);
75:D 2     end
76:D 2     else flushbuffer;
77:D 2     fleof := fpos + 1; fmodified := true;
78:D 2     if fleof>fpeof then
79:D 2     begin { move the physical end of file }
80:D 2       fpos := ((fleof * 256) div 256) * 256;
81:D 2       call(unitable^[funit].dam,fp^,funit,STRETCHIT);
82:D 2       if fleof>fpeof then begin ioresult := ord(ieof); escape(0); end;
83:D 2       fpos := fleof - 1;
84:D 2     end;
85:D 2     flastpos := bufend; bufparams;
86:D 2   end;
87:D 2 end;
88:S 2
89:D 2 procedure wnextbyte(c:char);
90:D 2 begin { append access only }
91:D 2   with fp^ do
92:D 2   begin
93:D 2     if fpos>=bufend then wendbuffer;
94:D 2     fbuffer[fpos - flastpos] := c; fbufchanged := true;
95:D 2     fpos := fpos + 1; fleof := fpos; fmodified := true;
96:D 2   end;
97:D 2   { wnextbyte }
98:D 2 end;
99:S 2
100:D 2 procedure writeendline;
101:D 2 var
102:D 2   tposit, j : integer;
103:D -8
104:D 2 begin
105:D 2   with fp^ do
106:D 2   begin
107:D 2     if freptcnt=0 then
108:D 2     begin { zero length record }
109:D 2       if odd(fpos) then wnextbyte(' ');
110:D 2       wnextbyte(chr(0)); wnextbyte(chr(0));
111:D 2     end
112:D 2     else { have some data }
113:D 2     begin
114:D 2       tposit := fpos; j := tposit - freptcnt - 2;
115:D 2       { rewrite the record size }
116:D 2       if j<flastpos then loadbuffer(j);
117:D 2       fbuffer[j - flastpos] := chr(freptcnt div 256);
118:D 2       fbuffer[(j+1) - flastpos] := chr(freptcnt mod 256);
119:D 2       fbufchanged := true;
120:D 2     end

```



```

121:C      5      end;
122:C      4      end;
123:C      3      end; { writeendline }
124:S
125:D      2      procedure wendfile;
126:C      3      begin
127:C      4      with fp^ do
128:C      5      begin
129:C      6      if freptcnt>0 then writeendline;
130:C      7      { write logical end of file marker }
131:C      8      if odd(fpos) then wnextbyte(' '); { pad to even position }
132:C      9      if ffeof<fpeof then
133:C      10     begin
134:C      11     wnextbyte(chr(255)); wnextbyte(chr(255));
135:C      12     end;
136:C      13     if fbufchanged then flushbuffer;
137:C      14     call(unitable^[funit].tm,fp,flush,ioreult,0,0);
138:C      15     end;
139:C      16     end;
140:S
141:D      2      function min(v1,v2,v3:integer):integer;
142:C      3      begin
143:C      4      if v1<v2 then
144:C      5      begin
145:C      6      { v1 or v3 }
146:C      7      if v1<v3 then min := v1 else min := v3;
147:C      8      end
148:C      9      else
149:C      10     { v2 or v3 }
150:C      11     begin
151:C      12     if v2<v3 then min := v2 else min := v3;
152:C      13     end;
153:C      14     end;
154:S
155:D      2      function rnextbyte:char;
156:C      3      begin
157:C      4      with fp^ do
158:C      5      begin
159:C      6      if fpos>=ffeof then rendfile;
160:C      7      if (fpos>=bufend) then loadbuffer(fpos);
161:C      8      rnextbyte := fbuffer[fpos - flastpos];
162:C      9      fpos := fpos + 1;
163:C      10     end;
164:C      11     end;
165:S
166:D      2      procedure getrecsize;
167:C      3      begin
168:C      4      with fp^ do
169:C      5      begin
170:C      6      if odd(fpos) then freptcnt := ord(rnextbyte);
171:C      7      freptcnt := ord(rnextbyte);
172:C      8      if freptcnt>127 then rendfile;
173:C      9      freptcnt := (freptcnt * 256) + ord(rnextbyte);
174:C      10     end;
175:C      11     end;
176:S
177:D      2      procedure readchars;
178:C      3      var
179:C      4      count, i : integer;
180:C      5      begin
181:C      6      { readchars }
182:C      7      with fp^ do
183:C      8      begin

```

```

181:C      4      i:=0;
182:C      5      if bufsize=1 then
183:C      6      begin { single character read }
184:C      7      if freptcnt=0 then
185:C      8      begin buffer[0] := ' '; freptcnt := -1; end
186:C      9      else
187:C      10     begin buffer[0] := rnextbyte; freptcnt := freptcnt - 1; end;
188:C      11     end;
189:C      12     { multi character read }
190:C      13     while i<bufsize do
191:C      14     begin
192:C      15     if freptcnt=0 then
193:C      16     begin { end of record }
194:C      17     buffer[i] := ' '; i := i + 1;
195:C      18     if i<bufsize then getrecsize
196:C      19     else freptcnt := -1;
197:C      20     end
198:C      21     else
199:C      22     begin { move data bytes }
200:C      23     if fpos>=ffeof then rendfile
201:C      24     else seekposit(fpos);
202:C      25     count := min(bufsize-i, bufend-fpos, freptcnt);
203:C      26     moveleft(fbuffer[fpos-flastpos],buffer[i],count);
204:C      27     i := i + count; freptcnt := freptcnt - count; fpos := fpos + count;
205:C      28     end;
206:C      29     { while }
207:C      30     feoln := (freptcnt<0);
208:C      31     if not feoln then fpos := -fpos;
209:C      32     end;
210:C      33     { readchars }
211:S
212:D      2      procedure readstring;
213:D      3      var i, count : integer;
214:C      4      begin
215:C      5      i := 0;
216:C      6      with fp^ do
217:C      7      begin
218:C      8      if freptcnt>0 then
219:C      9      while i<bufsize do
220:C      10     begin { read data bytes }
221:C      11     if fpos>=ffeof then rendfile else seekposit(fpos);
222:C      12     count := min(bufsize-i, bufend-fpos, freptcnt);
223:C      13     moveleft(fbuffer[fpos-flastpos],buffer[i],count);
224:C      14     freptcnt := freptcnt - count; i := i + count; fpos := fpos + count;
225:C      15     buffer[i] := chr(i);
226:C      16     if freptcnt=0 then i := bufsize
227:C      17     end;
228:C      18     fpos := -fpos;
229:C      19     end;
230:C      20     end;
231:S
232:D      2      procedure writechars;
233:D      3      var
234:D      4      i, count : integer;
235:S
236:C      5      begin
237:C      6      { writechars }
238:C      7      with fp^ do
239:C      8      begin
240:C      9      if (freptcnt = 0) and (bufsize>0) then

```



```

241:C      5      if odd(fpos) then wnextbyte(' '); { pad to even size }
242:C      5      wnextbyte(chr(255)); wnextbyte(chr(255)); { dummy count field }
243:C      5      end;
244:C      4      i:=0;
245:C      4      while i<bufsize do
246:C      5      begin { write data character(s) }
247:C      5      if fpos>=bufend then wendbuffer;
248:C      5      count := min(bufsize-i, bufend-fpos, 32767-freptcnt);
249:C      5      if count<=0 then { too many characters for the record }
250:C      5      begin ioreresult := ord(ibadformat); escape(0); end;
251:C      5      moveleft(buffer[i],fbuffer[fpos-flastpos],count);
252:C      5      fpos := fpos + count; freptcnt := freptcnt + count; i := i + count;
253:C      5      fbufchanged := true;
254:C      5      end; { while }
255:C      4      fleof := fpos;
256:C      4      end; { with }
257:C      3      end; { write chars }
258:S
259:C      2      begin { asciiam }
260:C      2      ioreresult:=0;
261:C      2      try
262:C      3      with fp^ do
263:C      4      begin
264:C      5      if flastpos<0 then { force buffer load }
265:C      5      begin flastpos := -buflength; fbufchanged := false; end;
266:C      5      bufparams; fpos := abs(position);
267:C      4      case request of
268:C      5      readbytes, readtoeol:
269:C      5      begin
270:C      5      if request=readtoeol then buffer[0] := chr(0);
271:C      5      if fbufchanged then
272:C      5      begin { close last record }
273:C      5      fpos := fleof; wendfile;
274:C      5      fpos := abs(position); { restore fpos }
275:C      5      end;
276:C      5      if position<0 then
277:C      5      begin { sequential read }
278:C      5      if request=readbytes then readchars
279:C      5      else readstring;
280:C      5      end { sequential read }
281:C      5      else
282:C      5      begin { positioned read }
283:C      5      if position>fp^.fleof then rendfile
284:C      5      else seekposit(position);
285:C      5      getrecsize;
286:C      5      if request=readbytes then readchars
287:C      5      else readstring;
288:C      5      end; { positioned read }
289:C      5      end;
290:C      5      writebytes:
291:C      5      begin
292:C      5      if position<>0 then writechars { normal write }
293:C      5      else
294:C      5      begin { rewrite }
295:C      5      flastpos := 0; bufparams; freptcnt := 0; writechars;
296:C      5      end;
297:C      5      end;
298:C      5      flush: if fbufchanged then
299:C      5      begin fpos := fleof; wendfile; end;
300:C      5      writeeol: { end the line }

```

```

301:C      5      begin
302:C      5      if position<>0 then writeendline
303:C      5      else
304:C      5      begin { zero length record at start of file }
305:C      5      flastpos := 0; bufparams; freptcnt := 0;
306:C      5      wnextbyte(chr(0)); wnextbyte(chr(0));
307:C      5      end;
308:C      5      end;
309:C      5      otherwise ioreresult := ord(ibadrequest);
310:C      5      end; { case }
311:C      4      end; { with }
312:C      3      recover
313:C      3      if escapecode<0 then escape(escapecode);
314:C      2      end; { asciiam }
315:C      1      end; { ascii module }
316:S
317:C      1      import asciimodule,sysglobals,loader;
318:C      1      begin { installascii }
319:C      1      amtable^[ASCIIFILE] := asciiam;
320:C      1      suffixtable^[ASCIIFILE] := 'ASC';
321:C      1      efttable^[ASCIIFILE] := 1;
322:C      1      markuser;
323:C      1      end. { rev'16 R }

```

No errors. No warnings.

***** Nonstandard language features enabled *****

BAT

Description

BAT provides the initialization for the battery backup (power fail) option.

Usage

BAT can read the date and time from the battery option, also set the timeout for power fail.

Requirements

SYSGLOBALS and SYSDEVS.

Notes

Also see module SYSDEVS.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1: S      (*
2: S
3: S      (c) Copyright Hewlett-Packard Company, 1983.
4: S      All rights are reserved. Copying or other
5: S      reproduction of this program except for archival
6: S      purposes is prohibited without the prior
7: S      written consent of Hewlett-Packard Company.
8: S
9: S
10: S      RESTRICTED RIGHTS LEGEND
11: S
12: S      Use, duplication, or disclosure by the Government
13: S      is subject to restrictions as set forth in
14: S      paragraph (b) (3) (B) of the Rights in Technical
15: S      Data and Computer Software clause in
16: S      DAR 7-104.9(a).
17: S
18: S      HEWLETT-PACKARD COMPANY
19: S      0 Fort Collins, Colorado      *)
20: S
21: S
22: S      0 $modcal$
23: S      0 $heap_dispose off$
24: S      0 $iocheck off$
25: S      0 $range off$ $ovflcheck off$
26: S      0 $debug off$
27: S      0 $STACKCHECK OFF$
28: S      0 $SEARCH 'INITLOAD','ASM','INIT','SYSDEVS'$
29: S
30: S      0 program initbat(OUTPUT);
31: S
32: S      1 module bat;
33: S      1 import sysglobals, sysdevs;
34: S      1 export
35: S      1 procedure batinit;
36: S
37: S      1 implement
38: S      1 type
39: S      1     statustype = packed record
40: S      1     case integer of
41: S      1     0: (pad1 : 0..63;
42: S      1     busy : boolean;
43: S      1     ready : boolean);
44: S      1     1: (statbyte : byte);
45: S      1     END;
46: S
47: S      1 var bat8041statusreg[4554785 { 458001 } ]: char;
48: S      1 bat8041cmdreg [4554785 { 458001 } ]: char;
49: S      1 bat8041datareg [4554753 { 458021 } ]: char;
50: S
51: S      1 procedure wait4batready;
52: S      -2 2 var batstatus: statustype;
53: S      2 begin
54: S      2 repeat
55: S      3 batstatus.statbyte:=ord(bat8041statusreg);
56: S      3 until not batstatus.busy;
57: S      2 end;
58: S
59: S      1 procedure wait4batreadready;
60: S      -2 2 var batstatus: statustype;

```

```

61: C      2 begin
62: C      2 repeat
63: C      3 batstatus.statbyte:=ord(bat8041statusreg);
64: C      3 until batstatus.ready;
65: C      2 end;
66: S
67: S      1 procedure dobatcommand(cmd: byte; numdata: integer; b1, b2, b3, b4, b5: byte);
68: S
69: S      2 procedure batdataout(d: byte);
70: S      3 begin wait4batready; bat8041datareg := chr(d); end;
71: S
72: S      2 begin
73: S      2 if batterypresent then
74: S      3 begin
75: S      3 wait4batready;
76: S      3 bat8041cmdreg := chr(cmd);
77: S      3 if numdata >= 1 then batdataout(b1);
78: S      3 if numdata >= 2 then batdataout(b2);
79: S      3 if numdata >= 3 then batdataout(b3);
80: S      3 if numdata >= 4 then batdataout(b4);
81: S      3 if numdata >= 5 then batdataout(b5);
82: S      3 end;
83: S      2 end;
84: S
85: S      1 procedure batreadbyte(var data: byte);
86: S      2 begin
87: S      2 wait4batreadready; data := ord(bat8041datareg);
88: S      2 end;
89: S
90: S      1 procedure batinit;
91: S      2 begin
92: S      2 batcommand(167,2,23,112,0,0,0); {set power fail to 60 seconds}
93: S      2 batcmdhook := dobatcommand;
94: S      2 batreadhook := batreadbyte;
95: S      2 end;
96: S
97: S      1 end;
98: S
99: S      1 import bat, loader, sysdevs;
100: S
101: S      1 begin
102: S      1 if batterypresent THEN
103: S      2 begin batinit; markuser; end;
104: S      1 end.
105: S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

BUBBLES

Description

BUBBLES is the TM for bubble memory mass storage.

Requirements

BUBDVR, SYSGLOBALS, IODECLARATIONS, and ISR.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1 S
2 S
3 S
4 S
5 S
6 S
7 S
8 S
9 S
10 S
11 S
12 S
13 S
14 S
15 S
16 S
17 S
18 S
19 S
20 S
21 S
22 S
23 S
24 S
25 S
26 S
27 S
28 S
29 S
30 S
31 S
32 S
33 S
34 S
35 S
36 S
37 S
38 S
39 S
40 S
41 S
42 S
43 S
44 S
45 S
46 S
47 S
48 S
49 S
50 S
51 S
52 S
53 S
54 S
55 S
56 S
57 S
58 S
59 S
60 S

```

(*)

(c) Copyright Hewlett-Packard Company, 1983.
All rights are reserved. Copying or other
reproduction of this program except for archival
purposes is prohibited without the prior
written consent of Hewlett-Packard Company.

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government
is subject to restrictions as set forth in
paragraph (b) (3) (B) of the Rights in Technical
Data and Computer Software clause in
DAR 7-104.9(a).

HEWLETT-PACKARD COMPANY
Fort Collins, Colorado (*)

```

0 $SYSPROGS
0 $DEBUG OFF$RANGE OFF$
0 program bubbles;
1 module bubble;
1 $SEARCH 'BUB_OVR'$
1 import bub_dvr,sysglobals,iodeclarations,isr;
1 export
1
1 procedure bub_tm(fp:fbp; request:amrequesttype;
2 anyvar buffer : window; bufsize, position : integer);
1
1 procedure bub_isr(isrib : pisrib);
1
1 procedure bub_init;
1
1 IMPLEMENT
1 procedure xlate_errors(error : berrortype);
2 begin
2 case error of
2 bnoerror : ioresult := ord(inoerror);
2 btimeout : ioresult := ord(zbadhardware);
2 bopfailed : ioresult := ord(zbadhardware);
2 bbadinterrupt: ioresult := ord(zstrange);
2 bbadsector : ioresult := ord(znosuchblk);
2 bbadcount : ioresult := ord(zbadmode);
2 bnotbubble : ioresult := ord(znodevice);
2 bbaddata : ioresult := ord(zbadblock);
2 biofail : ioresult := ord(ztimeout);
2 otherwise ioresult := ord(zcatchall);
2 end;
2
1 procedure bub_tm(fp:fbp; request:amrequesttype;
2 anyvar buffer : window; bufsize, position : integer);
2
2 var
2 card : anyptr;
2 info : infoptr;
2
2 begin
2 ioresult := ord(inoerror);

```

```

61 C
62 C
63 C
64 C
65 C
66 C
67 C
68 C
69 C
70 C
71 C
72 C
73 C
74 C
75 C
76 C
77 C
78 C
79 C
80 C
81 C
82 C
83 C
84 C
85 C
86 C
87 C
88 C
89 C
90 C
91 C
92 C
93 C
94 C
95 C
96 C
97 C
98 C
99 C
100 C
101 C
102 C
103 C
104 C
105 C
106 C
107 C
108 C
109 C
110 C
111 C
112 C
113 C
114 C
115 C
116 C
117 C
118 C
119 C
120 C

```

```

with fp^, unitable^[funit] do
begin
card := isc_table[sc].card_ptr;
if isc_table[sc].io_tmp_ptr = NIL then ioresult := ord(znodevice)
else
begin
info := addr(isc_table[sc].io_tmp_ptr^.drv_misc);
with info^ do
case request of
startread:
if runstate<>b_idle then ioresult := ord(znotready)
else
begin
if (position+bufsize)<fpeof then
begin
bbuffer.brec := addr(buffer);
bstart := fileid + byteoffset + position;
bcount := bufsize;
betry := 3;
if bcount<>0 then bubbread(card,info);
if request=writebytes then
while runstate<>b_idle do; { wait for idle }
if runstate=b_idle then xlate_errors(errorcode);
end
else ioresult := ord(ieof);
end;
startwrite,
writebytes:
if runstate<>b_idle then ioresult := ord(znotready)
else
begin
if (position+bufsize)<fpeof then
begin
bbuffer.brec := addr(buffer);
bstart := fileid + byteoffset + position;
if bufsize>0
then bcount := (bufsize + 255) DIV 256 * 256
else bcount := bufsize;
betry := 3;
if bcount<>0 then bubbwrite(card,info);
if request=writebytes then
while runstate<>b_idle do; { wait for idle }
if runstate=b_idle then xlate_errors(errorcode);
end
else ioresult := ord(ieof);
end;
flush; { NO_OP }
clearunit:
begin
bubdoreset(card,info);
xlate_errors(info^.errorcode);
if ioresult=ord(inoerror)
then umaxbytes := maxbytes { set device size }
else umaxbytes := 0;
end;
unitstatus:
begin
fbusy := runstate<>b_idle;
if not fbusy then xlate_errors(errorcode);
end;
otherwise ioresult := ord(ibadrequest);
end; { case }
end; { if nount }

```



```

121:C      3      end;      { with }
122:C      2      end; { bub_tm }
123:D      1 $DEBUG OFF$
124:D      1
125:D      2      procedure bub_isr(isrib : pisrib);
126:D      -4      2      var
127:D      -8      2      temps : pio_tmp_ptr;
128:C      2      info : info_ptr;
129:C      2      begin
130:C      2      temps := addr(isrib^);
131:C      2      info := addr(temps^.drv_misc);
132:C      2      bubble_isr(temps^.card_addr,info);
133:S      2      end; { bub_isr }
134:D      1      procedure bub_init;
135:D      2      const
136:D      2      intreg = 3;      { interrupt reg offset }
137:D      2      type
138:D      2      cardrec = packed record
139:D      2      pad : byte;
140:D      2      id : byte;
141:D      2      end;
142:D      2      var
143:D      -4      2      sc : integer;
144:D      -8      2      isrib : pisrib;
145:D      -12     2      info : info_ptr;
146:D      -16     2      card : ^cardrec;
147:C      2      begin
148:C      2      { scan selectcode table for bubble devices }
149:C      2      for sc:= iomisc to iomaxisc do
150:C      3      with isc_table[sc] do
151:C      4      begin
152:C      4      { fix isc table to work for system 2.0, 2.1 or 2.2 }
153:C      4      if ((card_type=1) and (card_id=0)) or
154:C      5      ((card_type=8) and (card_id=30)) then
155:C      5      begin
156:C      5      card := card_ptr;
157:C      5      if card^.id=30 then
158:C      6      begin
159:C      6      card_type := 8; { bubble memory card }
160:C      6      card_id := 30;
161:C      6      isrib := addr(io_tmp_ptr^);
162:C      6      info := addr(io_tmp_ptr^.drv_misc);
163:C      6      bubbleinfo(card,info);
164:C      6      if info^.errorcode=bnerror then
165:C      7      begin
166:C      7      permisrlink(bub_isr, { ISR PROC }
167:C      7      addr(card^,intreg), { INTERRUPT REG ADDRESS }
168:C      7      hex('CO'),hex('CO'), { MASK and VALUE }
169:C      7      info^.priority, { INTERRUPT PRIORITY }
170:C      7      isrib); { ISR INFO POINTER }
171:C      7      end;
172:C      6      end;
173:C      5      end;
174:C      4      end;
175:C      2      end; { bub_init }
176:D      1
177:C      1 end; { bub_ops }
178:D      1 { bubbles installation program }
179:D      1 import bubble,loader;
180:S

```

```

181:C      1 begin
182:C      1   bub_init;
183:C      1   markuser;
184:C      1 end.

```

No errors. No warnings.

***** Nonstandard language features enabled *****

C_HOOK

Description

C_HOOK initializes the graphics CRT state variables for the Series 200 Model 236C color computer and provides Model 236C dump graphics procedure.

Requirements

SYSDEVS and SYSGLOBALS.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:D 0 {
2:D 0 { HP 9836C graphics hooks, and workstation init module }
3:D 0 {
4:D 0 { Module = C_HOOK
5:D 0 { Programmer = BJS
6:D 0 { Date = 2-7-83
7:D 0 {
8:D 0 { Purpose: To initialize the graphics crt state variables for the HP 9836C. }
9:S
10:D 0 { Rev history
11:D 0 { Created - 02-07-83
12:D 0 { Modified - 01-18-84 jws
13:S
14:S ( (c) Copyright Hewlett-Packard Company, 1983.
15:S All rights are reserved. Copying or other
16:S reproduction of this program except for archival
17:S purposes is prohibited without the prior
18:S written consent of Hewlett-Packard Company.
19:S
20:S
21:S RESTRICTED RIGHTS LEGEND
22:S
23:S Use, duplication, or disclosure by the Government
24:S is subject to restrictions as set forth in
25:S paragraph (b) (3) (B) of the Rights in Technical
26:S Data and Computer Software clause in
27:S DAR 7-104.9(a).
28:S
29:S HEWLETT-PACKARD COMPANY
30:D 0 Fort Collins, Colorado }
31:S
32:D 0 $modcal$
33:D 0 program c_hook(output);
34:S
35:D 1 import sysdevs,sysglobals;
36:S
37:D 1 type
38:D 1 sysflag_def = packed record
39:D 1 bit7,bit6,hpib,crt_config,
40:D 1 kbd,high,big_graph,alpha50 : boolean;
41:D 1 end;
42:S
43:D 1 crt_reg_def = packed record
44:D 1 selfinit,bit14,bit13,top1,top2,highlite,
45:D 1 graph,alpha,
46:D 1 bit7,bit6,bit5,bit4,bit3,bit2,bit1,bit0 : boolean;
47:D 1 end;
48:S
49:D 1 c_map_def = packed array [0..15] of shortint;
50:S
51:D 1 var
52:D 1 my_sysflag [ hex('ffffd2') ] : sysflag_def;
53:D 1 crt_reg [ hex('51ffff') ] : crt_reg_def;
54:D 1 c_map [ hex('51fb00') ] : c_map_def;
55:D 1 g_on [ hex('51ffff') ] : shortint;
56:D -8 1 old_toggle_hook : procedure;
57:D -16 1 old_dump_hook : procedure;
58:D -16 1 graphics_base ['GRAPHICSBASE'] : integer;
59:S
60:D 1 procedure dump_c;

```

```

61:S
62:D 2 { Purpose: To dump 'C' bit map }
63:S
64:D 2 label 1;
65:S
66:D 2 const
67:D 2 gwidthb = 64;
68:D 2 gmaxheight = 512;
69:D 2 gbufferSize = gwidthb + 6;
70:S
71:D 2 type
72:D 2 gbyte = 0..255;
73:D 2 row_def = packed array [0..(512*390)-1] of gbyte;
74:S
75:D 2 var
76:D -4 2 row : ^row_def;
77:S
78:D -74 2 gbuffer : packed array [1..gbufferSize] of char;
79:D -82 2 i,j : integer;
80:D -86 2 index : integer;
81:D -90 2 bit_mask : integer;
82:D -94 2 result : integer;
83:S
84:C 2 begin
85:S
86:C 2 row := anyptr(graphics_base);
87:S
88:C 2 gbuffer[1] := chr(27); { escape sequence for graphics }
89:C 2 gbuffer[2] := 'x';
90:C 2 gbuffer[3] := 'b';
91:C 2 gbuffer[4] := '6';
92:C 2 gbuffer[5] := '4';
93:C 2 gbuffer[6] := 'W';
94:S
95:C 2 for j := 0 to 389 do
96:C 3 begin
97:C 4 for i := 0 to 63 do
98:C 5 begin
99:C 6 result := 0;
100:C 6 index := j*512+1*8;
101:C 6 bit_mask := 256;
102:C 6 for index := index to index+7 do
103:C 7 begin
104:C 8 bit_mask := bit_mask div 2;
105:C 8 if row^[index] <> 0 then result := bit_mask+result;
106:C 8 end;
107:C 8 gbuffer[i+7] := chr(result);
108:C 4 end;
109:C 4 write(gfiles[4]^,gbuffer:gwidthb+6);
110:C 4 if ioreult <> ord(1noerror) then goto 1;
111:C 3 end;
112:S
113:C 2 write(gfiles[4]^,#27*'B'); { terminate graphics sequence }
114:C 2
115:C 2 end;
116:S
117:D 1 procedure toggle_c;
118:S
119:C 2 begin
120:C 2 graphicstate := not graphicstate;

```



```
121:C      2   if graphicstate then g_on := 1
122:C      3   else      g_on := 0,
123:C      2 end;
124:C
125:C      1 procedure init_c;
126:C
127:C      2 var
128:C      -4  2   i : integer;
129:C
130:C      2 begin
131:C      2   g_on := 0;      { turn off graphics screen }
132:C      2   c_map[0] := -1;
133:C      2   for i := 1 to 15 do
134:C      3   c_map[i] := 0;      { init color map is b&w }
135:C      2   graphics_base := hex('520000');
136:C      2   togglegraphicshook := toggle_c;
137:C      2   dumpgraphicshook := dump_c;
138:C      2 end;
139:C
140:C      -16 1 $partial_eval on$ { check for a HP 9836C }
141:C      1 begin
142:C      1   if (currentcrt = alphas) and { jws 1/18/84 }
143:C      2   my_sysflag.big_graph and my_sysflag.crt_config and
144:C      2   (not crt_reg.top1) and (crt_reg.top2) then
145:C      2   init_c;
146:C      1 end.
```

No errors. No warnings.

***** Nonstandard language features enabled *****

CLOCK

Description

CLOCK reads and sets the time using the keyboard clock.

Requirements

SYSGLOBALS, ASM, MISC, and SYSDEVS.

Notes

See also module SYSDEVS, and SYSGLOBALS for definitions.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1983.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S      RESTRICTED RIGHTS LEGEND
11:S
12:S      Use, duplication, or disclosure by the Government
13:S      is subject to restrictions as set forth in
14:S      paragraph (b) (3) (B) of the Rights in Technical
15:S      Data and Computer Software clause in
16:S      DAR 7-104.9(a).
17:S
18:S      HEWLETT-PACKARD COMPANY
19:D      *) Fort Collins, Colorado
20:S
21:S
22:D      0 $mndcals
23:D      0 $heap dispose off$
24:D      0 $iocheck off$
25:D      0 $range off$ $ovflcheck off$
26:D      0 $debug off$
27:D      0 $stackcheck off$
28:D      0 $search 'INITLOAD','ASM','INIT','SYSDEVS'$
29:D
30:D      0 program clockinit;
31:D
32:D      1 module clock;
33:D      1 import sysglcbls, asm, misc, sysdevs;
34:D      1 export procedure initclock;
35:D
36:D      1 implement
37:D
38:D      1 type trickint = packed record
39:D      1         case integer of
40:D      1         0: ( ipart: integer );
41:D      1         1: ( byte3: byte;
42:D      1           byte2: byte;
43:D      1           byte1: byte;
44:D      1           byte0: byte );
45:D      1         end;
46:D
47:D      1 var boottype[-576]: shortint;
48:D
49:D      1 procedure dosysdate(var thedate: daterec);
50:D      -24 2 var yr,dd,mm,k,k1,k2: integer;
51:D      -32 1 time: rctime;
52:D      2 begin
53:D      2     CALL(CLOCKIOHOOK,CSET,LTIME);
54:D      2     k:=ltime,packeddate+1;
55:D      2     k1:= k*4-1;
56:D      2     yr:= k1 div 1461;
57:D      2     dd:= (k1-(1461*yr)+4) div 4;
58:D      2     k2:=(5*dd-3);
59:D      2     mm:=k2 div 153;
60:D      2     dd:= k2-153*mm;

```

```

61:C      2     dd:=(dd+5) div 5;
62:C      2     if mm<10 then mm:=mm+3
63:C      2     else
64:C      2     begin mm:=mm-9;yr:=yr+1; end;
65:C      2     with thedate do
66:C      2     begin
67:C      2     year:=yr mod 100;(to protect our file)
68:C      2     month:=mm;
69:C      2     day:=dd;
70:C      2     end;
71:C      2     end;
72:S
73:D      1 procedure dosystime(var thetime: timerec);
74:D      -4 1 var t: integer;
75:D      2 begin
76:D      2     t:=sysclock mod (24*360000);
77:D      2     with thetime do
78:D      2     begin
79:D      2     hour := t div 360000;
80:D      2     minute := (t-(hour*360000)) div 6000;
81:D      2     centisecond := t mod 6000;
82:D      2     end;
83:D      2     end;
84:S
85:D      -8 1 procedure setrtctime(thetime: rctime);
86:D      -8 const
87:D      -8 cnmdb7:=183;
88:D      -8 cnmdb40:=64;
89:D      -8 var
90:D      -16 t1,t2: trickint;
91:D      -24 TTIME: RTTIME;
92:D      2 begin
93:D      2     TTIME:=THETIME; CALL(CLOCKIOHOOK,CSET,TTIME);
94:D      2     t1.ipart := thetime.packeddate;
95:D      2     t2.ipart := thetime.packedtime;
96:D      2     batcommand(cnmdb7,5,t1.byte1,t1.byte0,t2.byte2,t2.byte1,t2.byte0);
97:D      2     batcommand(cnmdb40,0,0,0,0,0,0);
98:D      2     { setrtctime }
99:D      2     end;
100:S
101:D      1 procedure dosetsysdate(thedata: daterec);
102:D      -20 2 var ltime: rctime; yr,mth,dy: integer;
103:D      2 begin
104:D      2     CALL(CLOCKIOHOOK,CSET,LTIME);
105:D      2     with ltime,thedata do
106:D      2     begin
107:D      2     yr := year; mth := month; dy := day;
108:D      2     if mth>2 then mth:=mth-3
109:D      2     4 else begin mth:=mth+9; yr:=yr-1; end;
110:D      2     packeddate:=(1461* yr) div 4 + (153*mth+2) div 5+dy-1;
111:D      2     end;
112:D      2     setrtctime(ltime);
113:D      2     end;
114:S
115:D      1 procedure dosetsystime(thetime: timerec);
116:D      -20 2 var ltime: rctime; hr,min,ctsec: integer;
117:D      2 begin
118:D      2     CALL(CLOCKIOHOOK,CSET,LTIME);
119:D      2     with ltime, thetime do
120:D      2     begin
121:D      2     hr := hour; min := minute; ctsec := centisecond;

```



```

121:C      3      packedtime:=((hr*3600)+min*60)*100+ctsec;
122:C      3      end;
123:C      2      setrtctime(ltime);
124:C      2      end;
125:S
126:D      1      procedure inittime;
127:D      -8      2      var thetime:rtctime;
128:D      -8      2      const
129:D      -8      2      cmd41=65;           {65 hex to load timer output buffer with time}
130:D      -8      2      cmd77=247;       {F7 hex to load data buffer with first byte}
131:D      -8      2      cmdf6=246;       {F6 hex to load data buffer with second byte}
132:D      -8      2      cmdf5=245;       {F5 hex to load data buffer with third byte}
133:D      -8      2      cmdf4=244;       {F4 hex to load data buffer with fourth byte}
134:D      -8      2      cmdf3=243;       {F3 hex to load data buffer with fifth byte}
135:D      -8      2      cmdf2=242;       {F2 hex to load a letter 'B', or 'P', or 'H' for
136:D      -8      2      var t:trickint;   Basic, or Pascal, or HPL respectively}
137:D      -12     2      begin (inittime)
138:C      2      thetime.packedtime := 0;
139:C      2      thetime.packeddate := 0;
140:C      2      if batterypresent then
141:C      2      with t do
142:C      3      begin
143:C      4      setintlevel(2);
144:C      4
145:S      4      ipart := 0;
146:C      4      batcommand(cmd41,0,0,0,0,0,0);
147:C      4      batcommand(cmd77,0,0,0,0,0,0);   byte1 := batbytereceived;
148:C      4      batcommand(cmdf6,0,0,0,0,0,0);   byte0 := batbytereceived;
149:C      4      thetime.packeddate := ipart;
150:C
151:S      4      ipart := 0;
152:C      4      batcommand(cmdf5,0,0,0,0,0,0);   byte2 := batbytereceived;
153:C      4      batcommand(cmdf4,0,0,0,0,0,0);   byte1 := batbytereceived;
154:C      4      batcommand(cmdf3,0,0,0,0,0,0);   byte0 := batbytereceived;
155:C      4      thetime.packedtime := ipart;
156:C
157:S      4      setintlevel(0); {lower cpu int level}
158:C      4      end;
159:C      2      setrtctime(thetime);
160:C      2      end; { inittime }
161:C
162:S
163:D      1      PROCEDURE DOLOCKOPS(CMD:CLOCKFUNC; ANYVAR DATA: CLOCKDATA);
164:C      2      BEGIN
165:C      2      CASE CMD OF
166:C      3      CGETDATE: DOSYSDATE(DATA.DATETYPE);
167:C      3      CGETTIME: DOSYSTIME(DATA.TIMETYPE);
168:C      3      CSETDATE: DOSETSYSDATE(DATA.DATETYPE);
169:C      3      CSETTIME: DOSETSYTIME(DATA.TIMETYPE);
170:C      3      END;
171:C      2      END;
172:S
173:D      1      procedure initclock;
174:C      2      begin
175:C      2      if boottype = 0 {powerup} then inittime;
176:C      2      boottype := 18;
177:C      2      CLDCKREQ=DDK:=DOLOCKOPS;
178:C      2      end;
179:S
180:C      1      end; { module clock }

```

```

181:S
182:D      1      import clock, loader;
183:S
184:C      1      begin
185:C      1      initclock;
186:C      1      markuser;
187:C      1      end.
188:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

CONVERT

Description

CONVERT translates to or from a memory image of a WS1.0-format page of text when reading or writing any kind of text file.

Usage

CONVERT is used by the Compiler, Assembler, and Editor.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1: (*
2:
3: (c) Copyright Hewlett-Packard Company, 1983.
4: All rights are reserved. Copying or other
5: reproduction of this program except for archival
6: purposes is prohibited without the prior
7: written consent of Hewlett-Packard Company.
8:
9:
10: RESTRICTED RIGHTS LEGEND
11:
12: Use, duplication, or disclosure by the Government
13: is subject to restrictions as set forth in
14: paragraph (b) (3) (B) of the Rights in Technical
15: Data and Computer Software clause in
16: DAR 7-104.9(a).
17:
18: HEWLETT-PACKARD COMPANY
19: 0 Fort Collins, Colorado *)
20:
21:
22: 0 Smodcal$
23: 0 Sdebug off, range off, ovflcheck off, stackcheck off, ioccheck off$
24:
25: 0 module convert_text;
26:
27: 1 import sysglobals, misc, asm;
28:
29: 1 export
30:
31: 1 const pagesize = 1024;
32:
33: 1 type pagebuftype = packed array[0..pagesize-1] of char;
34:
35: 1 procedure any_to_UCSD(var T: text; anyvar pagebuffer: pagebuftype);
36: 1 procedure UCSD_to_any(anyvar pagebuffer: pagebuftype; var T: text);
37:
38: 1 implement
39:
40: 1 procedure any_to_UCSD(var T: text; anyvar pagebuffer: pagebuftype);
41: label 1;
42: 2 const strsize = 120; {arbitrary choice}
43: 2 type str = string[strsize];
44: 2 strptr = ^str;
45: 2 charptr = ^char;
46: 2 var i, j: integer; s: str; sp: strptr;
47: 2 var i, j: integer; s: str; sp: strptr;
48: 2 var i, j: integer; s: str; sp: strptr;
49: 2
50: 2 procedure ioccheck;
51: 2 begin
52: 2 if ioresult <> ord(ioerror) then
53: 2 begin
54: 2 if ioresult = ord(ioeof) then ioresult := ord(ioerror);
55: 2 goto 1;
56: 2 end;
57: 2 end;
58:
59: 2 begin
60: 2 i := 0;
61: 2 while (i < pagesize - strsize) do

```

```

62: 3 begin
63: 3 if endlines then
64: 3 begin
65: 3 sp := addr(pagebuffer[i], -1);
66: 3 read(T, sp);
67: 3 i := strlen(sp);
68: 3 charptr(sp)^ := eol; ioccheck;
69: 3 i := i + 1;
70: 3 end
71: 3 else
72: 3 begin
73: 3 read(T, s); ioccheck;
74: 3 moveleft(s[i], pagebuffer[i], strlen(s));
75: 3 i := i + strlen(s);
76: 3 endlines := eoln(T); ioccheck;
77: 3 if endlines then begin
78: 3 pagebuffer[i] := eol; i := i + 1;
79: 3 readln(T); ioccheck;
80: 3 end;
81: 3
82: 2 1: for i := 1 to pagesize-1 do pagebuffer[i] := nulchar;
83: 2 end;
84:
85: 1 $ioccheck on$
86:
87: 1 procedure UCSD_to_any(anyvar pagebuffer: pagebuftype; var T: text);
88: label 1;
89: 2 type pageptr = ^pagebuftype;
90: 2 var i, j: integer;
91: 2 c: char;
92: 2 begin
93: 2 try
94: 2 i := 0;
95: 2 repeat
96: 2 c := pagebuffer[i];
97: 2 if c = Chr(dle) then
98: 2 begin
99: 2 i := i + 1;
100: 2 if i = pagesize then begin write(T, c); goto 1; end
101: 2 else
102: 2 begin
103: 2 c := pagebuffer[i];
104: 2 if c > ' ' then write(T, ' 'ord(c)-ord(' '));
105: 2 i := i + 1;
106: 2 if i = pagesize then goto 1;
107: 2 c := pagebuffer[i];
108: 2 end;
109: 2 end;
110: 2 j := 1;
111: 2 while (c <> eol) and (c <> nulchar) do
112: 2 begin
113: 2 j := j + 1;
114: 2 if j = pagesize then c := nulchar
115: 2 else c := pagebuffer[j];
116: 2 end;
117: 2 write(T, pageptr(addr(pagebuffer[i]))^j-i);
118: 2 if c = nulchar then goto 1;
119: 2 writeln(T);
120: 2 i := j + 1;

```



```
121:C      4 until i = pagesize;
122:C      3 i;
123:C      3 recover if escapecode <> -10 then escape(escapecode);
124:C      2 end;
125:S
126:C      1 end. (module convert_text)
127:S
128:S
```

No errors. No warnings.

**** Nonstandard language features enabled ****

CRT

Description

CRT is called as a transfer method (TM) by the file system, and contains the debugger window handler.

Requirements

SYSGLOBALS, ASM, MISC, and SYSDEVS.

Notes

See also SYSDEVS.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```
1:      (*
2:      (c) Copyright Hewlett-Packard Company, 1984.
3:      All rights are reserved. Copying or other
4:      reproduction of this program except for archival
5:      purposes is prohibited without the prior
6:      written consent of Hewlett-Packard Company.
7:
8:
9:
10:     RESTRICTED RIGHTS LEGEND
11:
12:     Use, duplication, or disclosure by the Government
13:     is subject to restrictions as set forth in
14:     paragraph (b) (3) (B) of the Rights in Technical
15:     Data and Computer Software clause in
16:     DAR 7-104.9(a).
17:
18:     HEWLETT-PACKARD COMPANY
19:     0 Fort Collins, Colorado      *)
20:
21: 0 $UCSD$
22: 0 $modcal$
23: 0 $heap_dispose off$
24: 0 $iocheck off$
25: 0 $range off$ $ovfcheck off$
26: 0 $debug off$
27: 0 $stackcheck off$
28: 0 $search 'INITLOAD','ASM','INIT','SYSEVS'S
29:
30: 0 program initcrt;
31:
32: 1 module crt;
33: 1 import sysglobals, asm, misc, sysdevs;
34: 1 export
35: 1 function alphacrt: boolean;
36:
37: 1 implement
38: 1
39: 1 const
40: 1   minkana = 161;
41: 1   maxkana = 223;
42: 1   yenromlocation = 128; ( location of Yen symbol in old CRT rom )
43:
44: 1 type
45:
46: 1 kanatocrtlookuptype = packed array [minkana..maxkana] of 128..255;
47: 1 romtokanatype = packed array[#128..#238] of 0..255;
48:
49: 1 crtregtype = 0..15;
50: 1 crtcmdwrdr = packed record case integer of
51: 1   0: (topbyte, botbyte: byte);
52: 1   1: (longword: shortint);
53: 1   2: (p1,p2, textfield, softfield: boolean);
54: 1   end;
55:
56: 1 crtscreeen = array[0..maxint] of crtword;
57: 1 scrptr = ^crtscreeen;
58:
59:
60:
```

```
61: 1 const
62: 1   kanatocrtlookup = kanatocrtlookuptype [
63: 1     { code 161 } 129,130,131,132,133,134,135,
64: 1     { code 168 } 136,137,138,139,140,141,142,143,
65: 1     { code 176 } 144,145,146,147,148,149,150,151,
66: 1     { code 184 } 152,153,154,155,156,157,158,159,
67: 1     { code 192 } 160,161,162,163,164,165,166,167,
68: 1     { code 200 } 173,174,177,178,180,188,190,191,
69: 1     { code 208 } 224,225,226,227,228,229,230,231,
70: 1     { code 216 } 232,233,234,235,236,237,238,179 ];
71:
72: 1   romtokanamap = romtokanatype [ 92, 161, 162,
73: 1     163, 164, 165, 166, 167, 168, 169, 170, 171, 172,
74: 1     173, 174, 175, 176, 177, 178, 179, 180, 181, 182,
75: 1     183, 184, 185, 186, 187, 188, 189, 190, 191, 192,
76: 1     193, 194, 195, 196, 197, 198, 199, 188, 169, 170,
77: 1     171, 172, 200, 201, 175, 176, 202, 203, 223, 204,
78: 1     181, 182, 183, 184, 185, 186, 187, 205, 189, 206,
79: 1     207, 192, 193, 184, 195, 196, 197, 198, 199, 200,
80: 1     201, 202, 203, 204, 205, 206, 207, 208, 209, 210,
81: 1     211, 212, 213, 214, 215, 216, 217, 218, 219, 220,
82: 1     221, 222, 208, 209, 210, 211, 212, 213, 214,
83: 1     215, 216, 217, 218, 219, 220, 221, 222];
84:
85:
86: 1   b9826info=crtirec[
87: 1     width:80,height:24,
88: 1     crtmemaddr:$316808 ( + 416),
89: 1     crtcontroladdr:$341185,
90: 1     keybufferaddr:$320448 ( + 416),
91: 1     progstateinfoaddr:$320582 ( + 416),
92: 1     keybuffersize: 72,
93: 1     cricon:
94: 1     crtconsttype [114,80,76,7,26,10,25,25,0,14,76,13],
95: 1     right(FS):chr(28),
96: 1     left(BS):chr(8),
97: 1     down(LP):chr(10), up(US):chr(31),
98: 1     badch(?):chr(63),
99: 1     chardel(BS):chr(8),stop(DC3):chr(19),
100: 1     break(DLE):chr(16),
101: 1     flush(ACK):chr(6), eof(ETX):chr(3),
102: 1     altmode(ESC):chr(27),
103: 1     linedel(DEL):chr(127),
104: 1     backspace(BS):chr(8),
105: 1     etx:chr(3),prefix:chr(0),
106: 1     prefixed:014[14 of false],
107: 1     cursormask: 0, spare: 0];
108:
109:
110: 1   environc=envirom[miscinfo:crtirec[
111: 1     nobreak:false,
112: 1     stupid:false,
113: 1     slowterm:false,
114: 1     hasxycrt:true,
115: 1     haslcrt:true, (?)
116: 1     hasclock:true,
117: 1     canupscroll:true,
118: 1     candomscroll:true],
119: 1     crttype:0,
120: 1     crtctrl:crtirec[
```



```

121:D      1      rlf:chr(31);
122:D      1      ndfs:chr(28);
123:D      1      eraseeol:chr(9);
124:D      1      eraseeos:chr(11);
125:D      1      home:chr(1);
126:D      1      escape:chr(0);
127:D      1      backspace:chr(8);
128:D      1      fillcount:10;
129:D      1      clearscreen:chr(0);
130:D      1      clearline:chr(0);
131:D      1      prefixed:b9[9 of false];
132:D      1
133:D      1      crtinfo:crtirec[
134:D      1          width:50,height:24,
135:D      1          crtmemaddr:5316608,
136:D      1          crtcontroladdr:5308417,
137:D      1          keybufferaddr: 5319008,
138:D      1          progstateinfoaddr: 5318092,
139:D      1          keybuffersize: 42,
140:D      1          crtcon: crtconsttype [64,50,49,10,25,9,25,
141:D      1              25,0,11,74,11],
142:D      1          right(F5):chr(28),
143:D      1          left(B5):chr(8),
144:D      1          down(LF):chr(10), up(U5):chr(31),
145:D      1          badch(?):chr(63),
146:D      1          charde1(BS):chr(8),stop(DC3):chr(19),
147:D      1          break(DLE):chr(16),
148:D      1          flush(ACK):chr(6), eof(ETX):chr(3),
149:D      1          altmode(ESC):chr(27),
150:D      1          linedel(DEL):chr(127),
151:D      1          backspace(BS):chr(8),
152:D      1          etx:chr(3),prefix:chr(0),
153:D      1          prefixed:b14[14 of false],
154:D      1          cursormask: 0, spare: 0];
155:S
156:D      1 var
157:S
158:D      -4 1 lptr: script;
159:D      -8 1 screenwidth: integer;
160:D      -12 1 screenheight: integer;
161:S
162:S
163:D      -18 1 maxx,maxy,screensize:shortint;
164:D      -22 1 screen:script;
165:D      -24 1 defaulthighlight: shortint;
166:S
167:D      -26 1 highlight: shortint;
168:D      -27 1 hascolor: boolean;
169:D      -32 1 pm6845addrreg:char;
170:D      -36 1 pm6845comdreg:char;
171:D      -37 1 nomap: boolean;
172:S      ( 3.0 bug fix jws 3/20/84 )
173:D      -37 1 crtldreg[hex('51FFFE')]: packed record
174:D      -37 1     b15,b14,b13: boolean;
175:D      -37 1     colorinfo: (cinfo0, cinfo1, cinfo2, cinfo3);
176:D      -37 1     b10,b9,b8,b7,b6,b5,b4,b3,b2,b1,b0: boolean;
177:D      -37 1     end;
178:S
179:S
180:D      1 procedure dumpa;

```

```

181:D      2 label 1;
182:D      -8 2 var row, column:integer;
183:D      -9 2 c: char;
184:D      -110 2 line: string[100];
185:C      2 begin with syscom.crtinfo do
186:C      3 begin
187:C      3     setstrlen(line, width);
188:C      3     for row := 0 to height-1 do
189:C      4 begin
190:C      4     for column := 0 to width-1 do
191:C      5 begin
192:C      5     c := screen^[row*width+column].character;
193:C      5     if (c >= #128) and (c <= #238) then c := chr(romtokanamap[c]);
194:C      5     line[column+1] := c;
195:C      5     end;
196:C      4     column := width;
197:C      4     while (column > 1) and (line[column] = ' ') do column := column - 1;
198:C      4     writeln(gfiles[4]^, line[column]);
199:C      4     if ioresult <> ord(incoerror) then goto 1;
200:C      4     end;
201:C      3 end;
202:C      2 1: end;
203:S
204:D      1 procedure toggleg;
205:D      2 var gon [5439488(530000 HEX)]:shortint;
206:D      2 goff[5472256(538000 HEX)]:shortint;
207:D      2 gbase['GRAPHICSBASE']: ^shortint;
208:S
209:C      2 begin
210:C      2     graphicstate:=not graphicstate;
211:C      2     if graphicstate then gbase:=addr(gon);
212:C      3     else gbase:=addr(goff);
213:C      2     gbase^ := gbase^;
214:C      2 end;
215:S
216:D      1 procedure dumpg;
217:D      2 label 1;
218:D      2 const gheight = 300; gheightb = 390;
219:D      2 gwidth = 50; gwidthb = 64;
220:D      2 gbuffer:=gwidthb*6;
221:D      2 type gword=packed record
222:D      2     dummy,growbyte:char;
223:D      2     end;
224:D      2 gdotrow=packed array[1..gwidth] of gword;
225:D      2 type gmemtype = packed array [1..gheight] of gdotrow;
226:D      2 gmembtype = packed array [1..gheightb, 1..gwidthb] of char;
227:D      2 gmem = ^gmemtype;
228:D      2 gmemb = ^gmembtype;
229:D      2 var graphicsbase['GRAPHICSBASE']: anyptr;
230:D      2 gbuffer:packed array[1..gbufferize] of char;
231:D      -70 2 i,j,rows,bufferize,pindex:integer;
232:D      -91 2 busy:boolean;
233:C      2 begin
234:C      2     gbuffer[1]:=chr(esc) (escape sequence for graphics);
235:C      2     gbuffer[2]:= '*';
236:C      2     gbuffer[3]:= 'b';
237:C      2     gbuffer[6]:= 'L';
238:C      2     if sysflag.biggraphics then
239:C      3 begin
240:C      3     gbuffer[4]:= '6';

```



```

241:C      3      gbuffer[5]:='4';
242:C      3      rows := gheightb;
243:C      3      buffersize := gwidthb+6;
244:C      3      end;
245:C      3      else begin
246:C      3      gbuffer[4]:='5';
247:C      3      gbuffer[5]:='0';
248:C      3      rows := gheight;
249:C      3      buffersize := gwidth+6;
250:C      3      end;
251:C      3      for i:= 1 to rows do
252:C      3      begin
253:C      3      if sysflag.biggraphics then
254:C      4      for j:=1 to gwidthb do gbuffer[j+6]:=gmemb(graphicsbase)^[i,j] else
255:C      4      for j:=1 to gwidth do gbuffer[j+6]:=gmemb(graphicsbase)^[i,j].growbyte;
256:C      3      write(gfiles[4]^, gbuffer:buffersize);
257:C      3      if ioresult <> ord(inoerror) then goto 1;
258:C      3      end;
259:C      3      write(gfiles[4]^, #27*'rB'); {terminate graphics sequence};
260:C      1;
261:C      2      end;
262:S      2
263:D      1      procedure crtcommand(reg: crtregtype; data: byte);
264:C      2      begin
265:C      2      pm8845addrreg^ := chr(reg);
266:C      2      pm8845comdreg^ := chr(data);
267:C      2      end;
268:D      -37
269:D      1      procedure doupdatecursor;
270:D      -2      var cursaddr: crtcmdwr;
271:C      2      begin
272:C      2      cursaddr.longword:=integer(screen) mod 8192 div 2 + ypos*screenwidth+xpos;
273:C      2      cursaddr.textfield := alphastate;
274:C      2      cursaddr.softfield:=alphastate;
275:C      2      crtcommand(14, cursaddr.topbyte);
276:C      2      crtcommand(16, cursaddr.botbyte);
277:C      2      end;
278:S      2
279:D      1      procedure togglea;
280:D      -2      var lcursaddr:crtcmdwr;
281:S      2
282:C      2      begin
283:C      2      alphastate:=not(alphastate);
284:C      2      lcursaddr.longword:=integer(screen) mod 8192 div 2;
285:C      2      lcursaddr.textfield:=alphastate;
286:C      2      lcursaddr.softfield:=alphastate;
287:C      2      crtcommand(12, lcursaddr.topbyte);
288:C      2      crtcommand(13, lcursaddr.botbyte);
289:C      2      doupdatecursor;
290:C      2      end;
291:S      2
292:S      2
293:D      1      procedure getxy(var x,y: integer);
294:C      2      begin
295:C      2      x := xpos;      y := ypos;
296:C      2      end;
297:S      2
298:D      1      procedure setxy(x, y: shortint);
299:C      2      begin
300:C      2      if x>=screenwidth then xpos:=maxx

```

```

301:C      3      else if x<0 then xpos:=0
302:C      4      else xpos := x;
303:C      2      if y>=screenheight then ypos:=maxy
304:C      3      else if y<0 then ypos:=0
305:C      4      else ypos := y;
306:C      2      end;
307:S      2
308:D      1      procedure gotoxy(x,y: integer);
309:C      2      begin
310:C      2      setxy(x,y);
311:C      2      doupdatecursor;
312:C      2      end;
313:S      2
314:S      2
315:D      -4      1      procedure clear(number: shortint);
316:D      2      var x,y: shortint;
317:C      2      begin
318:C      2      x:=xpos; y:=ypos;
319:C      2      while number>0 do
320:C      3      begin
321:C      3      screen^[y*screenwidth+x].wholeword:= ord(' ');
322:C      3      number:=number-1;
323:C      4      if x<maxx then x:=x+1
324:C      4      else begin x:=0; if y<maxy then y:=y+1 end;
325:C      3      end;
326:C      2      end;
327:S      2
328:D      1      procedure scrollup;
329:D      -2      var i: shortint;
330:C      2      begin
331:C      2      moveleft(screen^[screenwidth(1, 0)],
332:C      2      screen^[0(0, 0)]);
333:C      2      (screensize-screenwidth)*2);
334:C      2      for i:=0 to maxx do
335:C      3      screen^[maxy*screenwidth+i].wholeword:=ord(' ');
336:C      2      end;
337:S      2
338:D      -2      1      procedure scrolldown;(new 4/30/81)
339:D      2      var i: shortint;
340:C      2      begin
341:C      2      moveright(screen^[0(0, 0)],
342:C      2      screen^[screenwidth(1, 0)]);
343:C      2      (screensize-screenwidth)*2);
344:C      2      for i:=0 to maxx do screen^[i].wholeword := ord(' ');
345:C      2      end;
346:S      2
347:S      2
348:S      2
349:S      2
350:D      1      function maptocrt(c:char):char;
351:S      2
352:D      2      const illegalchar = #223;
353:D      2      { char to disp for illegal internal codes; looks like hp }
354:D      2      procedure mapromextocrt;
355:D      3      const
356:D      3      minromex = 168; { lookup table ranges }
357:D      3      maxromex = 255;
358:D      3      type romexsettype = set of minromex..maxromex;
359:D      3      const romexset = romexsettype [168..172,175,176,179,181..187,189,192..222,255];
360:D      3      { legal Romex codes }

```



```

361:C      3 begin
362:C      3   if (ord(c) < 128) or (ord(c) in romexset)
363:C      4     or nomap then { 3.0 bug fix jws 3/20/84 }
364:C      4     maptochr:=c
365:C      4   else
366:C      4     if ord(c)=188 { ROMAN8 yen char } { jws 3/1/84 }
367:C      5     then maptochr:=chr(yenromlocation) { jws 3/1/84 }
368:C      5     else
369:C      5       maptochr:=illegalchar;
370:C      3 end;
371:S
372:S
373:D      2 procedure mapkanatocrt ;
374:S
375:D      3 const yencode = 92;
376:S
377:S
378:S      { Converts Katakana codes to their correct "old" CRT rom location codes; also,
379:S      converts "illegal" Kana chars to the "hp" char. Note that the Yen symbol
380:D      3 overlays the USASCII backslash (\), and that code 255 is left unconverted. }
381:S
382:S
383:C      3 begin
384:C      3   if nomap then maptochr:=c
385:C      4   else begin
386:C      4     if ord(c) = yencode then maptochr := chr(yenromlocation)
387:C      5     else if (ord(c) < 128) or (ord(c) = 255) then maptochr:= c
388:C      6     else begin
389:C      6       if (ord(c) < minkana) or (ord(c) > maxkana) then maptochr := illegalchar
390:C      7       else maptochr := chr(kanatocrtlookup(ord(c)));
391:C      6     end;
392:C      4   end;
393:C      3 end; { mapkanatocrt }
394:S
395:S
396:C      2 begin
397:C      2   if kbdlang = katakana_kbd then mapkanatocrt
398:C      3   else mapromextocrt;
399:C      2 end;
400:S
401:S
402:D      1 procedure docrtio(fp: fbp; request: amrequesttype; anyvar buffer: window;
403:D      2                                     length, position: integer);
404:D      2 var c: char;
405:D      2   s: string[1];
406:D      2   buf: charptr;
407:C      2 begin
408:C      2   ioresult := ord(inoerror);
409:C      2   buf := addr(buffer);
410:C      2   case request of
411:C      3     {uwait: ; }
412:C      3     setcursor: gotoxy(fp^.fxpos, fp^.fypos);
413:C      3     getcursor: getxy (fp^.fxpos, fp^.fypos);
414:C      3     flush: {do nothing};
415:C      3     unitstatus: kbdio(fp, request, buffer, length, position);
416:C      3     clearunit: highlight := defaulthighlight;
417:C      3     readtoeol:
418:C      3       begin
419:C      3         buf := addr(buf^, 1);
420:C      3         buffer[0] := chr(0);

```

```

421:C      3   while length>0 do
422:C      4     begin
423:C      4       kbdio(fp, readtoeol, s, 1, 0);
424:C      4       if strien(s)=0 then length := 0
425:C      5       { else if s[1] = chr(etx) then length := 0 }
426:C      5       else begin
427:C      5         length := length - 1;
428:C      5         crtio(fp, writebytes, s[1], 1, 0);
429:C      5         buf := addr(buf^, 1);
430:C      5         buffer[0] := chr(ord(buffer[0])+1);
431:C      5         end;
432:C      4       end;
433:C      3     end;
434:C      3   startread;
435:C      3   readbytes:
436:C      3   begin
437:C      3     while length>0 do
438:C      4       begin
439:C      4         kbdio(fp, readbytes, buf^, 1, 0);
440:C      4         if buf^ = chr(etx) then length := 0
441:C      5         else length := length - 1;
442:C      4         if buf^ = eol then crtio(fp, writeeol, buf^, 1, 0);
443:C      5         else crtio(fp, writebytes, buf^, 1, 0);
444:C      4         buf := addr(buf^, 1);
445:C      4         end;
446:C      3       if request = startread then call(fp^.feot, fp);
447:C      3       end;
448:C      3       writeeol: begin
449:C      3         if ypos=maxy then scrollup;
450:C      3         gotoxy(0, ypos+1);
451:C      3         end;
452:C      3       startwrite;
453:C      3       writebytes:
454:C      3       begin
455:C      3         while length>0 do
456:C      4         begin
457:C      4         c:=buf^, buf:=addr(buf^,1); length:=length-1;
458:C      4         end;
459:C      3         case c of
460:C      4           homechar: setxy(0,0);
461:C      5           leftchar:  if (xpos = 0) and (ypos>0) then setxy(maxx, ypos-1)
462:C      5                     else setxy(xpos-1, ypos);
463:C      5           rightchar: if (xpos = maxx) and (ypos<maxy) then setxy(0, ypos+1)
464:C      5                     else setxy(xpos+1, ypos);
465:C      5           upchar:   begin if ypos <= 1 then scrollup;
466:C      5                     if ypos>0 then setxy(xpos, ypos-1);
467:C      5                     end;
468:C      5           downchar: if ypos=maxy then scrollup
469:C      5                     else setxy(xpos, ypos+1);
470:C      5           bellchar: beep;
471:C      5           cteos:   clear(screenwidth-(ypos*screenwidth+xpos));
472:C      5           cteol:   clear(screenwidth-xpos);
473:C      5           clearscr: begin setxy(0,0); clear(screenwidth); end;
474:C      5           eol:     setxy(0, ypos);
475:C      5           chr(etx): length:=0;
476:C      5           otherwise if (ord(c)>=128) and (ord(c)< 144) then
477:C      6             if hascolor then
478:C      7               if ord(c) >= 136 then highlight :=
479:C      8                 highlight mod 2048 + (ord(c)-136)*4096
480:C      8               else highlight :=

```



```

481:C      8      (highlight div 2048 * 8 + (ord(c)-128))*256
482:C      8      else highlight := (ord(c)-128)*256
483:C      7      else with screen^[ypos*screenwidth+xpos] do
484:C      7      begin
485:C      7      wholeword:=highlight+ ord(maptocrt(c));
486:C      7      if xpos = maxx then
487:C      6      begin
488:C      6      if ypos = maxy then scrollup;
489:C      6      setxy(x, ypos+1);
490:C      6      end
491:C      6      else setxy(xpos+1, ypos);
492:C      7      end;
493:C      5      end;
494:C      4      douppdatecursor;
495:C      4      end; {while}
496:C      3      if request = startwrite then call(fp^ feet, fp);
497:C      3      end;
498:C      3      otherwise ioreult := ord(lbadrequest);
499:C      3      end; {case}
500:C      2      end;
501:C      1
502:C      1      procedure lineops(op: crtlops; anyvar position: integer; c:char);
503:C      2      var
504:C      2      i: integer;
505:C      -8      2      sptr: ^string255;
506:C      2
507:C      2      begin
508:C      2      case op of
509:C      3      cllput: lptr^[position].wholeword:=ord(maptocrt(c));
510:C      3
511:C      3      cllshiftl:
512:C      3      begin
513:C      3      for i:=0 to (maxx-8) do lptr^[i]:=lptr^[i+1];
514:C      3      lptr^[maxx-8].wholeword:=ord(' ');
515:C      3      end;
516:C      3
517:C      3      cllshiftr:
518:C      3      begin
519:C      3      for i:=0 to (maxx-9) do lptr^[maxx-8-i]:=lptr^[maxx-9-i];
520:C      3      lptr^[0].wholeword:=ord(' ');
521:C      3      end;
522:C      3
523:C      3      cllclear:
524:C      3      for i:=0 to (maxx-8) do lptr^[i].wholeword:=ord(' ');
525:C      3
526:C      3      clldisplay:
527:C      3      begin
528:C      3      sptr:=addr(position);
529:C      3      for i:=1 to length(sptr^) do
530:C      4      lptr^[i-1].wholeword:=ord(maptocrt(sptr^[i]));
531:C      3      for i:=length(sptr^) to (maxx-8) do
532:C      4      lptr^[i].wholeword:=ord(' ');
533:C      3      end;
534:C      3
535:C      3      putstatus:
536:C      3      begin { position should be in range 0..7 }
537:C      3      lptr^[maxx-7+position].wholeword:=ord(c);
538:C      3      end;
539:C      3      end; { case}
540:C

```

```

541:C      2      end; { lineops }
542:C
543:C      1      procedure crtdebug(op: dbcrtops; var dbrec: dbcinfor );
544:C      2
545:C      2      type lptr = ^iarray;
546:C      2      iarray = array[0..maxint] of shortint;
547:C      2
548:C      2      var
549:C      -4      2      xtemp, ytemp: shortint;
550:C      -10      2      i,j,k: shortint;
551:C      -12      2      len: shortint;
552:C      -14      2      inc: shortint;
553:C      -174      2      temp: array[0..79] of shortint;
554:C      2
555:C      2      begin
556:C      2      with dbrec do begin
557:C      3      case op of
558:C      4      dbinfo: savesize:=(xmax-xmin+1)*(ymax-ymin+1)*2;
559:C      4
560:C      4      dbgotoxy:
561:C      4      begin
562:C      4      xtemp:=xpos; ytemp:=ypos;
563:C      4      xpos:=curx; ypos:=cury;
564:C      4      douppdatecursor;
565:C      4      xpos:=xtemp; ypos:=ytemp;
566:C      4      end;
567:C      4
568:C      4      dbscrollup,dbscrollldn:
569:C      4      begin
570:C      4      len:=(xmax-xmin+1)*2;
571:C      4      if op=dbscrollup then begin
572:C      5      j:=ymin;
573:C      5      inc:=screenwidth;
574:C      5      end
575:C      5      else begin
576:C      5      j:=ymax;
577:C      5      inc:=-screenwidth;
578:C      5      end;
579:C      5      j:=j*screenwidth+xmin;
580:C      5      for i:=(ymin+1) to ymax do begin
581:C      6      k:=j; j:=j+inc;
582:C      6      moveleft(screen^[j], screen^[k], len);
583:C      6      end;
584:C      5      for i:=0 to (xmax-xmin) do
585:C      6      screen^[j+1].wholeword:=ord(' ');
586:C      5      end;
587:C      4
588:C      4      dbscrollrl,dbscrollrr:
589:C      4      begin
590:C      4      len:=(xmax-xmin+1)*2-2; { fixed 4/13/84 }
591:C      4      j:=(ymin-1)*screenwidth+xmin;
592:C      4      if op=dbscrollrl then begin
593:C      5      j:=j+1;
594:C      5      k:=xmax-xmin-1;
595:C      5      end
596:C      5      else begin
597:C      6      k:=0;

```



```

601:C      5      end;
602:C      4      for i:=ymin to ymax do begin
603:C      5          j:=screenwidth;
604:C      5          if op=dbscroll then
605:C      6              moveleft(screen^[j],screen^[j-1], len)
606:C      6          else
607:C      6              moveright(screen^[j],screen^[j+1], len);
608:C      5          screen^[j+k].wholeword:=ord(' ');
609:C      5      end;
610:C      4      end;
611:S
612:C      4      dbhighl:
613:C      4      begin
614:C      4          i:=cursy*screenwidth+cursx;
615:C      4          screen^[i].wholeword:=ord(screen^[i].character)+(ord(c)-128)*256;
616:C      4      end;
617:S
618:S
619:C      4      dbput: screen^[cursy*screenwidth+cursx].wholeword:=ord(c);
620:S
621:C      4      dbclear:
622:C      5      for i:=ymin to ymax do
623:C      5          for j:=xmin to xmax do
624:C      6              screen^[i*screenwidth+j].wholeword:=ord(' ');
625:S
626:C      4      dbcline:
627:C      5      for i:=cursx to xmax do
628:C      5          screen^[cursy*screenwidth+i].wholeword:=ord(' ');
629:S
630:C      4      dbinit:
631:C      4      begin
632:C      4          for i:=0 to (savesize div 2)-1 do
633:C      5              iptr(savearea)^[i]:=ord(' ');
634:C      4          cursx:=xmin; cursy:=ymin;
635:C      4          areaisdbort:=true;
636:C      4      end;
637:S
638:C      4      dbexcg:
639:C      4      begin
640:C      4          k:=xmax-xmin+1;
641:C      4          for i:=ymin to ymax do begin
642:C      5              moveleft(screen^[i*screenwidth+xmin], temp, k*2);
643:C      5              moveleft(iptr(savearea)^[i-ymin]*k],
644:C      5                  screen^[i*screenwidth+xmin], k*2);
645:C      5              moveleft(temp, iptr(savearea)^[i-ymin]*k], k*2);
646:C      5          end;
647:C      4          if areaisdbort then begin
648:C      5              xtemp:=xpos; ytemp:=ypos;
649:C      5              xpos:=cursx; ypos:=cursy;
650:C      5              doupdatecursor;
651:C      5              xpos:=xtemp; ypos:=ytemp;
652:C      5          end
653:C      5          else doupdatecursor;
654:C      4          areaisdbort:=not areaisdbort;
655:C      4      end;
656:S
657:S
658:C      4      end; { of case }
659:C      3      end; { of with }
660:C

```

```

661:C      2      end; { procedure crtdebug }
662:S
663:S
664:C      -10     1      procedure alphacrtinit;
665:C      2          var cursaddr: crtcmdwrd; i,k: integer;
666:C      2          begin
667:C      3              with syscom^.crtinfo do
668:C      3              begin
669:C      3                  screer:=anyptr(crtmemaddr);
670:C      3                  screerwidth:=width;
671:C      3                  screerheight:=height;
672:C      3                  maxx:=width-1;
673:C      3                  maxy:=height-1;
674:C      3                  screensize:=width*height;
675:S
676:C      3          for i:=0 to screensize-1 do screen^[i].wholeword:=ord(' '); {clear screen}
677:C      3          pm6845addrreg:=anyptr(crtcontroladdr);
678:C      3          pm6845cmdreg:=anyptr(crtcontroladdr+2);
679:C      3          cursaddr.longword:=integer(screen) mod 8192 div 2;
680:C      3          cursaddr.textfield:=alphastate;
681:C      3          cursaddr.softfield:=alphastate;
682:C      3          crtcommand(12, cursaddr.topbyte);
683:C      3          crtcommand(13, cursaddr.botbyte);
684:C      3          defaulthighlight := 0; highlight := 0;
685:S
686:C      3          idle:=250;
687:C      3          nomap:=false;
688:C      3          if sysflag.crtconfigreg then begin
689:C      4              if crtldreg.b13 then begin { 3.0 bug jws 3/20/84 }
690:C      5                  nomap:=true; { 3.0 bug jws 3/20/84 }
691:C      5                  idle:=245; { 3.0 bug jws 3/20/84 }
692:C      5                  end; { 3.0 bug jws 3/20/84 }
693:C      4              hascolor := crtldreg.colorinfo > cinfo0;
694:C      4          end
695:C      3          else hascolor := false;
696:S
697:C      3          gotoxy(0,0);
698:C      3          dumpalphahook := dumpa;
699:C      3          dumpgraphicshook := dumpg;
700:C      3          togglealphahook := togglea;
701:C      3          togglegraphicshook := toggleg;
702:C      3          updatecursorhook:=doupdatecursor;
703:C      3          crtinfohook:=docrtic;
704:C      3          crtllhook:=lineops;
705:C      3          dbcrthook:=crtdebug;
706:C      3          crtinihook:=alphacrtinit;
707:C      3          lptr:=anyptr(keybufferaddr);
708:C      3          keybuffer^.maxsize:=maxx-8;
709:C      3          currentcrt:=alphatype;
710:C      2          end;
711:C      -37     1      function alphacrt:boolean;
712:C      2          var i[hex('512000')]:shortint;
713:C      2          j: shortint;
714:C      -2      2          begin
715:C      3              alphacrt:=true; { assume we have alpha screen }
716:C      3          try
717:C      3              j:=i; { attempt read from alpha screen ram }
718:C      3              syscom^.environc; { setup for my kind of environment }
719:C      3              if not sysflag.alpha80 then syscom^.crtinfo:=b9826info;
720:C

```



```
721:C      2  alphacrtinit;
722:C      3  recover
723:C      3  if escapecode=-12 then
724:C      4      alphacrt:=false { bus error -- no alpha screen }
725:C      4      else escape(escapecode);
726:C      2  end;
727:S
728:C      1  end; { of module }
729:S
730:S
731:D      1  import crt, loader;
732:S
733:C      1  begin
734:C      1      if alphacrt then markuser;
735:C      1  end.
736:S
```

No errors. No warnings.

***** Nonstandard language features enabled *****

Description

CS80 is a transfer method (TM) for all supported Command Set '80 and Subset '80 discs.

Usage

CS80 supports the following CS/80 discs:

- 7908
- 7911
- 7912
- 7914
- 7935
- 7941
- 7944 (tape drive only; no disc)
- 7945

CS80 supports the following SS/80 discs:

- 9133
- 9134

Requirements

DISCHPIB, DRVASM, IODECLARATIONS and DISC_INTF

Optionally: DMA

Notes

CS80 does support the 7933 discs (see AMIGO).

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

118:D 1 $page$
119:D 1
120:D 1
121:D 1 evu_type = (encoded volume/unit (1 byte) - status & copy commands)
122:D 1 packed record case integer of
123:D 1 0: (vvvv: unsgn4; {volume number}
124:D 1 uuuu: unsgn4); {unit number}
125:D 1 1: (evu_byte: signed8); {for -1 test}
126:D 1 end;
127:D 1
128:D 1 errorbit_type = (error bit assignments in status & status mask)
129:D 1 (
130:D 1 { (REJECT ERRORS FIELD)
131:D 1 { 0} eb0,
132:D 1 { 1} eb1,
133:D 1 { 2} channel_parity_error,
134:D 1 { 3} rns,
135:D 1 { 4} eb4,
136:D 1 { 5} illegal_opcode,
137:D 1 { 6} module_addressing,
138:D 1 { 7} address_bounds,
139:D 1 { 8} parameter_bounds,
140:D 1 { 9} illegal_parameter,
141:D 1 {10} message_sequence,
142:D 1 {11} eb1,
143:D 1 {12} message_length,
144:D 1 {13} eb13,
145:D 1 {14} eb14,
146:D 1 {15} eb15,
147:D 1 { (FAULT ERRORS FIELD)
148:D 1 {16} eb16,
149:D 1 {17} cross_unit,
150:D 1 {18} eb18,
151:D 1 {19} controller_fault,
152:D 1 {20} eb20,
153:D 1 {21} eb21,
154:D 1 {22} unit_fault,
155:D 1 {23} eb23,
156:D 1 {24} diagnostic_result,
157:D 1 {25} eb25,
158:D 1 {26} operator_release_required,
159:D 1 {27} diagnostic_release_required,
160:D 1 {28} internal_maintenance_required,
161:D 1 {29} eb29,
162:D 1 {30} power_fail,
163:D 1 {31} retransmit,
164:D 1 { (ACCESS ERRORS FIELD)
165:D 1 {32} illegal_parallel_operation,
166:D 1 {33} uninitialized_media,
167:D 1 {34} no_spare_available,
168:D 1 {35} not_ready,
169:D 1 {36} write_protect,
170:D 1 {37} no_data_found,
171:D 1 {38} eb38,
172:D 1 {39} eb39,
173:D 1 {40} unrecoverable_data_overflow,
174:D 1 {41} unrecoverable_data,
175:D 1 {42} eb42,
176:D 1 {43} end_of_file,
177:D 1 {44} end_of_volume,

```

```

178:D 1 {45} eb45,
179:D 1 {46} eb46,
180:D 1 {47} eb47,
181:D 1 { (INFORMATION ERRORS FIELD)
182:D 1 {48} operator_request,
183:D 1 {49} diagnostic_request,
184:D 1 {50} internal_maintenance_request,
185:D 1 {51} media_wear,
186:D 1 {52} latency_induced,
187:D 1 {53} eb53,
188:D 1 {54} eb54,
189:D 1 {55} auto_sparing_invoked,
190:D 1 {56} eb56,
191:D 1 {57} recoverable_data_overflow,
192:D 1 {58} marginal_data,
193:D 1 {59} recoverable_data,
194:D 1 {60} eb60,
195:D 1 {61} maintenance_track_overflow,
196:D 1 {62} eb62,
197:D 1 {63} eb63
198:D 1 );
199:D 1
200:D 1
201:D 1 status_mask_type =
202:D 1 packed array[errorbit_type] of boolean;
203:D 1
204:D 1
205:D 1 status_type =
206:D 1 packed record
207:D 1 { (IDENTIFICATION FIELD)
208:D 1 current_vu: evu_type; {current volume/unit}
209:D 1 requesting_unit: signed8; {unit requesting service}
210:D 1 { (ERROR REPORTING FIELDS)
211:D 1 errorbits: status_mask_type;
212:D 1 { (PARAMETER FIELD)
213:D 1 case integer of
214:D 1 {positive cases correspond to error bits}
215:D 1 -1: (nta: sva_type; {new target address}
216:D 1 faultlog: integer); {fault log}
217:D 1 -2: (aaa: sva_type; {affected area address}
218:D 1 afl: integer); {affected field length}
219:D 1 17: (uee: packed array[1..8] of signed8); {units experiencing errors}
220:D 1 24: (drr: packed array[1..8] of unsgn8); {diagnostic results}
221:D 1 38: (ta: sva_type); {target address}
222:D 1 41: (bba: sva_type); {bad block address}
223:D 1 48..50: (urr: packed array[1..8] of signed8); {units requesting release}
224:D 1 58: (btbs: sva_type); {block to be spared}
225:D 1 59: (rba: sva_type); {recoverable block address}
226:D 1 end; {case}

```



```

227:D 1 $page$
228:D 1
229:D 1 t_type = {'T' parameter in SET_RELEASE}
230:D 1 ( allow_release_timeout; (power-on default)
231:D 1 suppress_release_timeout);
232:D 1
233:D 1
234:D 1 z_type = {'Z' parameter in SET_RELEASE}
235:D 1 (disable_auto_release; (power-on default)
236:D 1 enable_auto_release);
237:S 1
238:D 1
239:D 1 CMD_type = {enumerated opcodes for device commands}
240:D 1 (CMDlocate_and_read, CMD1, CMDlocate_and_wrt, CMD3,
241:D 1 CMDlocate_and_ver, CMD5, CMDspare_block, CMD7,
242:D 1 CMDcopy_data, CMD9, CMDcold_load_read, CMD11,
243:D 1 CMD12, CMDrequest_status, CMDrelease, CMDrelease_denied,
244:D 1 CMDset_address_1V, CMDset_address_3V, CMDset_block_disp, CMD19,
245:D 1 CMD20, CMD21, CMD22, CMD23,
246:D 1 CMDset_length, CMD25, CMD26, CMD27,
247:D 1 CMD28, CMD29, CMD30, CMD31,
248:D 1 CMDset_unit_0, CMDset_unit_1, CMDset_unit_2, CMDset_unit_3,
249:D 1 CMDset_unit_4, CMDset_unit_5, CMDset_unit_6, CMDset_unit_7,
250:D 1 CMDset_unit_8, CMDset_unit_9, CMDset_unit_10, CMDset_unit_11,
251:D 1 CMDset_unit_12, CMDset_unit_13, CMDset_unit_14, CMDset_unit_15,
252:D 1 CMDinit_util_NEM, CMDinit_util_REM, CMDinit_util_SEM, CMDinit_diagnostic,
253:D 1 CMDno_op, CMDdescribe, CMD54, CMDinit_media,
254:D 1 CMDset_options, CMDset_rps, CMDset_retry_time, CMDset_release,
255:D 1 CMDset_burst_180, CMDset_burst_ABT, CMDset_status_mask, CMD63,
256:D 1 CMDset_vol_0, CMDset_vol_1, CMDset_vol_2, CMDset_vol_3,
257:D 1 CMDset_vol_4, CMDset_vol_5, CMDset_vol_6, CMDset_vol_7,
258:D 1 CMDset_retadd_mode, CMDwrite_file_mark, CMDunload, CMD75,
259:D 1 CMD76, CMD77, CMD78, CMD79,
260:D 1 CMD80, CMD81, CMD82, CMD83,
261:D 1 CMD84, CMD85, CMD86, CMD87,
262:D 1 CMD88, CMD89, CMD90, CMD91,
263:D 1 CMD92, CMD93, CMD94, CMD95,
264:D 1 CMD96, CMD97, CMD98, CMD99,
265:D 1 CMD100, CMD101, CMD102, CMD103,
266:D 1 CMD104, CMD105, CMD106, CMD107,
267:D 1 CMD108, CMD109, CMD110, CMD111,
268:D 1 CMD112, CMD113, CMD114, CMD115,
269:D 1 CMD116, CMD117, CMD118, CMD119,
270:D 1 CMD120, CMD121, CMD122, CMD123,
271:D 1 CMD124, CMD125, CMD126, CMD127,
272:D 1 CMD128 {the field width is forced to 8 bits for packing considerations} );
273:D 1
274:D 1 const
275:D 1 transparent_sec = 18;
276:D 1 command_sec = 5;
277:D 1 execution_sec = 14;
278:D 1 reporting_sec = 16;

```

```

279:D 1 $page$
280:D 1
281:D 1 errorbits_owning_parmfield = {errorbits which set the parameter field}
282:D 1 [
283:D 1 (REJECT ERRORS FIELD)
284:D 1 { 0 } eb0, {unknown, but assumed}
285:D 1 { 1 } eb1, {unknown, but assumed}
286:D 1 { 3 } eb3, {unknown, but assumed}
287:D 1 { 4 } eb4, {unknown, but assumed}
288:D 1 { 11 } eb11, {unknown, but assumed}
289:D 1 { 13 } eb13, {unknown, but assumed}
290:D 1 { 14 } eb14, {unknown, but assumed}
291:D 1 { 15 } eb15, {unknown, but assumed}
292:D 1 (FAULT ERRORS FIELD)
293:D 1 { 16 } eb16, {unknown, but assumed}
294:D 1 { 17 } cross_unit, {unknown, but assumed}
295:D 1 { 18 } eb18, {unknown, but assumed}
296:D 1 { 20 } eb20, {unknown, but assumed}
297:D 1 { 21 } eb21, {unknown, but assumed}
298:D 1 { 23 } eb23, {unknown, but assumed}
299:D 1 { 24 } diagnostic_result, {unknown, but assumed}
300:D 1 { 25 } eb25, {unknown, but assumed}
301:D 1 { 29 } eb29, {unknown, but assumed}
302:D 1 (ACCESS ERRORS FIELD)
303:D 1 { 38 } eb38, {unknown, but assumed}
304:D 1 { 39 } eb39, {unknown, but assumed}
305:D 1 { 41 } unrecoverable_data, {unknown, but assumed}
306:D 1 { 42 } eb42, {unknown, but assumed}
307:D 1 { 45 } eb45, {unknown, but assumed}
308:D 1 { 46 } eb46, {unknown, but assumed}
309:D 1 { 47 } eb47, {unknown, but assumed}
310:D 1 (INFORMATION ERRORS FIELD)
311:D 1 { 48 } operator_request, {unknown, but assumed}
312:D 1 { 49 } diagnostic_request, {unknown, but assumed}
313:D 1 { 50 } internal_maintenance_request, {unknown, but assumed}
314:D 1 { 53 } eb53, {unknown, but assumed}
315:D 1 { 54 } eb54, {unknown, but assumed}
316:D 1 { 56 } eb56, {unknown, but assumed}
317:D 1 { 58 } marginal_data, {unknown, but assumed}
318:D 1 { 59 } recoverable_data, {unknown, but assumed}
319:D 1 { 60 } eb60, {unknown, but assumed}
320:D 1 { 62 } eb62, {unknown, but assumed}
321:D 1 { 63 } eb63, {unknown, but assumed}
322:D 1 ];
323:D 1
324:S 1
325:D 1 errorbits_requesting_release = {errorbits which request release}
326:D 1 [
327:D 1 { 48 } operator_request,
328:D 1 { 49 } diagnostic_request,
329:D 1 { 50 } internal_maintenance_request
330:D 1 ];

```



```

331:D      1 $page$
332:S      1
333:S      {
334:S          NOTE: the following functions each perform a COMPLETE transaction. They:
335:S              . issue a (device or transparent) command      (Command message)
336:S              . transfer data if applicable                    (Execution message)
337:S              . return the resulting QSTAT                     (Reporting message)
338:D      1 }
339:D      1 function chan_indep_clr (uep: uep_type): unsgn8;
340:D      1 function set_unit      (uep: uep_type; unit: unsgn4): unsgn8;
341:D      1 function set_unitvol   (uep: uep_type): unsgn8;
342:D      1 function status       (uep: uep_type; var status_bytes: status_type): unsgn8;
343:D      1 function release      (uep: uep_type; unit: unsgn4): unsgn8;
344:D      1 function describe     (uep: uep_type; var describe_bytes: describe_type): unsgn8;
345:D      1 function set_release   (uep: uep_type; t: t_type; z: z_type): unsgn8;
346:D      1 function set_options   (uep: uep_type; options_byte: unsgn8): unsgn8;
347:D      1 function set_status_mask (uep: uep_type; status_mask: status_mask_type): unsgn8;
348:S      1
349:S      {
350:S          NOTE: The following routines do not, in themselves, perform a complete
351:S          transaction. They provide some of the messages necessary for
352:S          transactions which are broken apart to allow overlapped transfers.
353:S      1 }
354:D      1
355:D      1 procedure ICuvalc (uep: uep_type, address, len: integer; cmd: CMD_type);
356:D      1 function qstat      (uep: uep_type): unsgn8;
357:S      1
358:D      1 implement (CS80)
359:S      1
360:S      1
361:D      1 var
362:D      -20 1 most_recent_status: status_type; {for post-mortem diagnostic purposes only!!}
363:S      1
364:S      1
365:D      1 function qstat(uep: uep_type): unsgn8;
366:S      1 {
367:S          receive a REPORTING message
368:S          return the QSTAT byte
369:D      2 }
370:D      2 var
371:D      2 qstat_byte: {the 1 byte in the reporting message}
372:D      2 packed record
373:D      2   b: unsgn8;
374:D      -2 2 end;
375:D      2
376:D      2 begin {qstat}
377:D      2   HPI8short_msge_in(uep, reporting_sec, addr(qstat_byte), sizeof(qstat_byte));
378:D      2   qstat := qstat_byte.b;
379:D      2 end; {qstat}

```

```

379:D      -20 1 $page$
380:S      1
381:D      1 function chan_indep_clr(uep: uep_type): unsgn8;
382:S      1 {
383:S          issue the CHANNEL_INDEPENDENT_CLEAR command
384:S          return the QSTAT byte
385:D      2 }
386:D      2 var
387:D      2 cic: {the 2 bytes in the channel independent clear command message}
388:D      2 packed record
389:D      2   setunit: CMD_type;
390:D      2   ci_clr: unsgn8;
391:D      -2 2 end;
392:D      2
393:D      2 begin {chan_indep_clr}
394:D      2   cic.setunit := CMD_type(signed16(CMDset_unit_0)+uep^.du);
395:D      2   cic.ci_clr := 8;
396:D      2   HPI8short_msge_out(uep, transparent_sec, addr(cic), sizeof(cic));
397:D      2   HPI8wait_for_ppol(uep);
398:D      2   chan_indep_clr := qstat(uep);
399:D      2 end; {chan_indep_clr}
400:S      1
401:D      1 procedure ICc(uep: uep_type; cmd: CMD_type);
402:S      1 {
403:S          issue the specified command
404:D      2 }
405:D      2 var
406:D      2 c: {the 1-byte command message}
407:D      2 packed record
408:D      2   cmd: CMD_type;
409:D      -2 2 end;
410:D      2
411:D      2 begin {ICc}
412:D      2   c.cmd := cmd;
413:D      2   HPI8short_msge_out(uep, command_sec, addr(c), sizeof(c));
414:D      2 end; {ICc}
415:D      -20 1
416:D      -20 1
417:D      1 procedure ICuc(uep: uep_type; unit: unsgn4; cmd: CMD_type);
418:S      1 {
419:S          issue the specified SET_UNIT & command
420:D      2 }
421:D      2 var
422:D      2 uc: {the 2-byte command message}
423:D      2 packed record
424:D      2   setunit: CMD_type;
425:D      2   cmd: CMD_type;
426:D      -2 2 end;
427:D      2
428:D      2 begin {ICuc}
429:D      2   uc.setunit := CMD_type(signed16(CMDset_unit_0)+unit);
430:D      2   uc.cmd := cmd;
431:D      2   HPI8short_msge_out(uep, command_sec, addr(uc), sizeof(uc));
432:D      2 end; {ICuc}

```



```

431:D -20 1 $page$
432:D -20 1
433:D 1 function set_unit(uep: uep_type; unit: unsn4): unsn8;
434:S {
435:S     issue the SET_UNIT command
436:S     return the QSTAT byte
437:D 2
438:C 2
439:C 2 begin {set_unit}
440:C 2     ICc(uep, CMD_type(signed16(CMDset_unit_0)+unit));
441:C 2     HPIBwait_for_ppol(uep);
442:C 2     set_unit := qstat(uep);
443:C 2 end; {set_unit}
444:S
445:D 1 function set_unitvol(uep: uep_type): unsn8;
446:S {
447:S     issue the SET_UNIT & SET_VOLUME commands
448:S     return the QSTAT byte
449:D 2
450:C 2
451:C 2 begin {set_unitvol}
452:C 2     ICc(uep, uep^.du, CMD_type(signed16(CMDset_vol_0)+uep^.dv));
453:C 2     HPIBwait_for_ppol(uep);
454:C 2     set_unitvol := qstat(uep);
455:C 2 end; {set_unitvol}
456:S
457:D 1 function status(uep: uep_type; var status_bytes: status_type): unsn8;
458:S {
459:S     issue the REQUEST_STATUS command
460:S     place the 20 bytes of status in the passed variable 'status_bytes'
461:S     return the QSTAT byte
462:D 2
463:C 2
464:C 2 begin {status}
465:C 2     ICc(uep, CMDrequest_status);
466:C 2     HPIBwait_for_ppol(uep);
467:C 2     HPIBshort_msge_in(uep, execution_sec, addr(status_bytes), sizeof(status_bytes));
468:C 2     most_recent_status := status_bytes; {for post-mortem diagnostic purposes only!!!}
469:C 2     HPIBwait_for_ppol(uep);
470:C 2     status := qstat(uep);
471:C 2 end; {status}
472:D -20 1
473:D -20 1 function release(uep: uep_type; unit: unsn4): unsn8;
474:S {
475:S     SET_UNIT & issue the RELEASE command
476:S     return the QSTAT byte
477:D 2
478:C 2
479:C 2 begin {release}
480:C 2     ICc(uep, unit, CMDrelease);
481:C 2     HPIBwait_for_ppol(uep);
482:C 2     release := qstat(uep);
483:C 2 end; {release}

```

```

483:D -20 1 $page$
484:S
485:D 1 function describe(uep: uep_type; var describe_bytes: describe_type): unsn8;
486:S {
487:S     issue the DESCRIBE command
488:S     place the 37 bytes of description in the passed variable 'describe_bytes'
489:S     return the QSTAT byte
490:D 2
491:C 2
492:C 2 begin {describe}
493:C 2     ICc(uep, CMDdescribe);
494:C 2     HPIBwait_for_ppol(uep);
495:C 2     HPIBshort_msge_in(uep, execution_sec, addr(describe_bytes), sizeof(describe_bytes));
496:C 2     HPIBwait_for_ppol(uep);
497:C 2     describe := qstat(uep);
498:C 2 end; {describe}
499:S
500:D 1 function set_release(uep: uep_type; t: t_type; z: z_type): unsn8;
501:S var
502:D 2 sr: {the 3 bytes in the SET_UNIT & SET_RELEASE command message}
503:D 2 packed record
504:D 2     setunit: CMD_type;
505:D 2     setrel: CMD_type;
506:D 2     tbit: t_type;
507:D 2     zbit: z_type;
508:D 2     pad: 0..63;
509:D 2 end;
510:D -4
511:C 2
512:C 2 begin {set_release}
513:C 2     sr.setunit := CMDset_unit_15; {always addressed to the controller}
514:C 2     sr.setrel := CMDset_release;
515:C 2     sr.tbit := t;
516:C 2     sr.zbit := z;
517:C 2     sr.pad := 0;
518:C 2     HPIBshort_msge_out(uep, command_sec, addr(sr), sizeof(sr));
519:C 2     HPIBwait_for_ppol(uep);
520:C 2     set_release := qstat(uep);
521:C 2 end; {set_release}
522:S
523:D 1 function set_options(uep: uep_type; options_byte: unsn8): unsn8;
524:S var
525:D 2 so: {the 2 bytes in the SET_OPTIONS command message}
526:D 2 packed record
527:D 2     setoptin: CMD_type;
528:D 2     ob: unsn8;
529:D 2 end;
530:C 2
531:C 2 begin {set_options}
532:C 2     so.setoptin := CMDset_options;
533:C 2     so.ob := options_byte;
534:C 2     HPIBshort_msge_out(uep, command_sec, addr(so), sizeof(so));
535:C 2     HPIBwait_for_ppol(uep);
536:C 2     set_options := qstat(uep);
537:C 2 end; {set_options}

```



```

536:D -20 1 $page$
537:S
538:D -8 1 function set_status_mask(uep: uep_type; status_mask: status_mask_type): uns8;
539:D -8 2 var
540:D -8 3 ssm: (the 10 bytes in the SET_STATUS_MASK command message)
541:D -8 3 packed record
542:D -8 3 3 nop: CMD_type;
543:D -8 3 3 setsmask: CMD_type;
544:D -8 3 3 stsmask: status_mask_type;
545:D -18 3 end;
546:C
547:C 2 begin (set_status_mask)
548:C 2 3 ssm.nop := CMDno_op;
549:C 2 3 ssm.setsmask := CMDset_status_mask;
550:C 2 3 ssm.stsmask := status_mask;
551:C 2 3 HPBshort_msge_out(uep, command_sec, addr(ssm), sizeof(ssm));
552:C 2 3 HPBwait_for_ppol(uep);
553:C 2 3 set_status_mask := qstat(uep);
554:C 2 end; (set_status_mask)
555:S
556:D
557:D 1 procedure lCuvalc(uep: uep_type; address, len: integer; cmd: CMD_type);
558:D 2 {
559:D 2 3 issue the following command sequence:
560:D 2 3 3 SET_UNIT (u)
561:D 2 3 3 SET_VOLUME (v)
562:D 2 3 3 SET_ADDRESS (a)
563:D 2 3 3 SET_LENGTH (l)
564:D 2 3 3 specified COMMAND (c)
565:D 2 3 }
566:D 2 3 var
567:D 2 3 3 uvalc: (the 17 bytes in the command message)
568:D 2 3 3 packed record
569:D 2 3 3 3 setunit: CMD_type; {set unit}
570:D 2 3 3 3 setvol: CMD_type; {set volume}
571:D 2 3 3 3 nop1: CMD_type; {nop}
572:D 2 3 3 3 setadd: CMD_type; {set address}
573:D 2 3 3 3 sva: sva_type; {single vector address}
574:D 2 3 3 3 nop2: CMD_type; {nop}
575:D 2 3 3 3 setlen: CMD_type; {set length}
576:D 2 3 3 3 length: integer; {length}
577:D 2 3 3 3 cmd: CMD_type; {specified command}
578:D -18 2 3 end;
579:C 2 3 begin (lCuvalc)
580:C 2 3 3 uvalc.setunit := CMD_type(signed16(CMDset_unit_0)+uep^.du);
581:C 2 3 3 uvalc.setvol := CMD_type(signed16(CMDset_vol_0)+uep^.dv);
582:C 2 3 3 uvalc.nop1 := CMDno_op;
583:C 2 3 3 uvalc.setadd := CMDset_address_1V;
584:C 2 3 3 uvalc.sva.utb := 0;
585:C 2 3 3 uvalc.sva.lfb := address;
586:C 2 3 3 uvalc.nop2 := CMDno_op;
587:C 2 3 3 uvalc.setlen := CMDset_length;
588:C 2 3 3 uvalc.length := len;
589:C 2 3 3 uvalc.cmd := cmd;
590:C 2 3 HPBshort_msge_out(uep, command_sec, addr(uvalc), sizeof(uvalc));
591:C 2 3 end; (lCuvalc)
592:C 2 end; (CS80)

```

```

593:D 1 $page$
594:S
595:D 1 module CS80dsr; {Command Set '80 Driver Support Routines}
596:S
597:D 1 import
598:D 1 3 sysglobals, bkgnd, tapebuf, CS80.
599:S
600:D 1 export
601:S
602:D 1 3 procedure invalidate_stateinfo(uep: uep_type);
603:D 1
604:D 1 3 procedure handle_bad_status(uep: uep_type; ok_to_config: boolean; var retry_required: boolean);
605:D 1
606:D 1 3 procedure configure(uep: uep_type);
607:S
608:D 1 implement (CS80dsr)
609:S
610:D
611:D 1 procedure invalidate_stateinfo(uep: uep_type);
612:D 2 var
613:D -2 2 3 lun: unitnum;
614:D -6 2 3 scanner_uep: uep_type;
615:C 2 3 begin (invalidate_stateinfo)
616:C 2 3 3 for lun := 1 to maxunit do
617:C 2 3 3 3 begin
618:C 2 3 3 3 3 scanner_uep := addr(unitable^[lun]);
619:C 2 3 3 3 3 if (scanner_uep^.letter='Q') and
620:C 2 3 3 3 3 3 (scanner_uep^.sc = uep^.sc) and
621:C 2 3 3 3 3 3 (scanner_uep^.ba = uep^.ba) and
622:C 2 3 3 3 3 3 (scanner_uep^.du = uep^.du)
623:C 2 3 3 3 3 3 3 (don't qualify dv because all volumes ARE affected!)
624:C 2 3 3 3 3 3 3 then (invalidate all CS80 state info!)
625:C 2 3 3 3 3 3 3 3 begin
626:C 2 3 3 3 3 3 3 3 3 scanner_uep^.umediavalid := false; {media possibly changed}
627:C 2 3 3 3 3 3 3 3 3 scanner_uep^.dvrtmp2 := -1; {block size possibly changed!}
628:C 2 3 3 3 3 3 3 3 3 if scanner_uep=tapebuf_uep then
629:C 2 3 3 3 3 3 3 3 3 3 tapebuf_state := undefined;
630:C 2 3 3 3 3 3 3 3 3 end; (if)
631:C 2 3 3 3 3 3 3 3 end; (for)
632:C 2 3 end; (invalidate_stateinfo)

```



```

633:D      1 $page$
634:S
635:D      1 procedure handle_bad_status(uep: uep_type; ok_to_config: boolean; var retry_required: boolean);
636:D      2
637:D      2   var
638:D      2     iorval_to_report: iorsltwd; {to hold the first reportable error}
639:D      2     working_iorval: iorsltwd; {cleared each time status is read}
640:D      2     status_bytes: status_type;
641:D      2     eb_scan, parameter_field_owner: errorbit_type;
642:D      2     reConfiguration_needed: boolean;
643:D      2
644:C      2   begin (handle_bad_status)
645:C      2
646:C      2     iorval_to_report := inoerror;
647:C      2
648:C      2     repeat
649:C      3
650:C      3       if status(uep, status_bytes) <> 0 then
651:C      4         ioresc_bkgnd(uep, zbadhardware);
652:C      3
653:C      3       working_iorval := inoerror;
654:C      3       parameter_field_owner := channel_parity_error; {doesn't REALLY own it!}
655:C      3       reConfiguration_needed := false;
656:C      3
657:C      3       for eb_scan := eb63 downto eb0 do
658:C      4         if status_bytes.errorbits[eb_scan] then
659:C      5           begin
660:C      5
661:C      5             if eb_scan in errorbits_owning_parmfield then
662:C      6               parameter_field_owner := eb_scan;
663:C      5
664:C      5             case eb_scan of
665:C      6
666:C      6               {specific fatal errors}
667:C      6               channel_parity_error,
668:C      6               controller_fault,
669:C      6               unit_fault,
670:C      6               diagnostic_result:
671:C      6                 working_iorval := zbadhardware;
672:C      6               illegal_opcode,
673:C      6               parameter_bounds,
674:C      6               illegal_parameter:
675:C      6                 working_iorval := zbadmode; {some cmds optional in SS/80}
676:C      6               module_addressing:
677:C      6                 working_iorval := znodvice;
678:C      6               address_bounds,
679:C      6               end_of_volume:
680:C      6                 working_iorval := znosuchblk;
681:C      6               uninitialized_media:
682:C      6                 if status_bytes.errorbits[power_fail]
683:C      7                   then {probably an uncertified tape; allow access anyway}
684:C      6                   else working_iorval := zuninitialized;
685:C      6               no_spare_available:
686:C      6                 working_iorval := zinitfail;
687:C      6               not_ready:
688:C      6                 working_iorval := znoready;
689:C      6               write_protect:
690:C      6                 working_iorval := zprotected;
691:C      6               no_data_found,
692:C      6               end_of_file:

```

```

693:C      6       working_iorval := znoblock;
694:C      6       unrecoverable_data_overflow,
695:C      6       unrecoverable_data:
696:C      6       working_iorval := zbadblock;
697:S
698:C      6   {power fail}
699:C      6   power_fail:
700:C      6     begin
701:C      6       invalidate_stateinfo(uep);
702:C      6       if uep.unreportchange then
703:C      7         working_iorval := zmediumchanged;
704:C      6       reConfiguration_needed := true;
705:C      6       retry_required := true;
706:C      6     end;
707:S
708:C      6   {retryable errors}
709:C      6   operator_release_required,
710:C      6   diagnostic_release_required,
711:C      6   internal_maintenance_required,
712:C      6   retransmit:
713:C      6     retry_required := true;
714:S
715:C      6   {errors indicating release requested}
716:C      6   operator_request,
717:C      6   diagnostic_request,
718:C      6   internal_maintenance_request:
719:C      6     {do nothing here; release below if parameter field owned};
720:S
721:C      6   {errors indicating reconfiguration needed}
722:C      6   media_wear, {supposed to be masked out}
723:C      6   latency_induced, {supposed to be masked out}
724:C      6   eb53, {supposed to be masked out}
725:C      6   eb54, {supposed to be masked out}
726:C      6   auto_sparing_invoked, {supposed to be masked out}
727:C      6   eb56, {supposed to be masked out}
728:C      6   recoverable_data_overflow, {supposed to be masked out}
729:C      6   marginal_data, {supposed to be masked out}
730:C      6   recoverable_data, {supposed to be masked out}
731:C      6   eb60, {supposed to be masked out}
732:C      6   maintenance_track_overflow, {supposed to be masked out}
733:C      6   eb62, {supposed to be masked out}
734:C      6   eb63, {supposed to be masked out}
735:C      6     reConfiguration_needed := true;
736:S
737:C      6   {errors not covered by the above cases}
738:C      6   otherwise
739:C      6     { specifically including:
740:C      6       message_sequence,
741:C      6       message_length,
742:C      6       cross_unit,
743:C      6       illegal_parallel_operation }
744:C      6     working_iorval := zcatchall;
745:C      6
746:C      6   end; {case}
747:C      6
748:C      5   end; {if}

```



```

749:C      3 $page$
750:S
751:C      3      if iorval_to_report=inoerror then (none previously found; report this one)
752:C      4      iorval_to_report := working_iorval; (it can be inoerror also!)
753:C      3
754:C      3      if parameter_field_owner in errorbits_requesting_release then
755:C      4      if not (status_bytes.urr[1] in [0..15]) then
756:C      5      ioresc_bkgnd(uep, zcatchall)
757:C      6      else if release(uep, status_bytes.urr[1])<>0 then
758:C      6      (handle the bad qstat elsewhere; worry not, the device won't forget!);
759:S
760:C      3      if reconfiguration_needed and ok_to_config then
761:C      4      configure(uep);
762:C      3
763:C      3      until set_unit(uep, status_bytes.current_vu.rrrr)=0; (restore original command unit)
764:C      3
765:C      3      if iorval_to_report<>inoerror then
766:C      4      ioresc_bkgnd(uep, iorval_to_report);
767:C      3
768:C      3      end; (handle_bad_status)
769:S
770:S
771:D      1 procedure configure(uep: uep_type);
772:D      2
773:D      2      var
774:D      -1      escape_caught: boolean;
775:D      -2      saved_uerrorchange: boolean;
776:D      -3      retry_required: boolean;
777:D      -4      describe_bytes: describe_type;
778:D      -40      bcd_prod_num: (within the describe bytes)
779:D      -40      packed_record case integer of
780:D      -40      0: (dn: unsngn24);
781:D      -40      1: (bcd: packed array[1..6] of unsngn4);
782:D      -44      end;
783:D      -44      fixed_volume_byte:
784:D      -44      packed_record case integer of
785:D      -44      0: (b: unsngn8);
786:D      -44      1: (bit: packed array[0..7] of boolean);
787:D      -46      end;
788:D      -48      prod_num: signed16;
789:D      -50      index: signed16;
790:D      -50
791:D      -50      const
792:D      -50      masked = true;
793:D      -50      unmasked = false;
794:D      -50
795:D      -50      my_status_mask = status_mask_type
796:D      -50      [
797:D      -50      (REJECT ERRORS FIELD)
798:D      -50      {
799:D      -50      0 eb0:
800:D      -50      1 eb1:
801:D      -50      2 channel_parity_error:
802:D      -50      3 eb3:
803:D      -50      4 eb4:
804:D      -50      5 illegal_opcode:
805:D      -50      6 module_addressing:
806:D      -50      7 address_bounds:
807:D      -50      8 parameter_bounds:
808:D      -50      9 illegal_parameter:
809:D      -50      10 message_sequence:

```

```

809:D      -50      11 eb11:
810:D      -50      12 message_length:
811:D      -50      13 eb13:
812:D      -50      14 eb14:
813:D      -50      15 eb15:
814:D      -50      (FAULT ERRORS FIELD)
815:D      -50      {
816:D      -50      16 eb16:
817:D      -50      17 cross_unit:
818:D      -50      18 eb18:
819:D      -50      19 controller_fault:
820:D      -50      20 eb20:
821:D      -50      21 eb21:
822:D      -50      22 unit_fault:
823:D      -50      23 eb23:
824:D      -50      24 diagnostic_result:
825:D      -50      25 eb25:
826:D      -50      26 operator_release_required:
827:D      -50      27 diagnostic_release_required:
828:D      -50      28 internal_maintenance_required:
829:D      -50      29 eb29:
830:D      -50      30 power_fail:
831:D      -50      31 retransmit:
832:D      -50      (ACCESS ERRORS FIELD)
833:D      -50      {
834:D      -50      32 illegal_parallel_operator:
835:D      -50      33 uninitialized_media:
836:D      -50      34 no_spare_available:
837:D      -50      35 not_ready:
838:D      -50      36 write_protect:
839:D      -50      37 no_data_found:
840:D      -50      38 eb38:
841:D      -50      39 eb39:
842:D      -50      40 unrecoverable_data_overflow:
843:D      -50      41 unrecoverable_data:
844:D      -50      42 eb42:
845:D      -50      43 end_of_file:
846:D      -50      44 end_of_volume:
847:D      -50      45 eb45:
848:D      -50      46 eb46:
849:D      -50      47 eb47:
850:D      -50      (INFORMATION ERRORS FIELD)
851:D      -50      {
852:D      -50      48 operator_request:
853:D      -50      49 diagnostic_request:
854:D      -50      50 internal_maintenance_request:
855:D      -50      51 media_wear:
856:D      -50      52 latency_induced:
857:D      -50      53 eb53:
858:D      -50      54 eb54:
859:D      -50      55 auto_sparing_invoked:
860:D      -50      56 eb56:
861:D      -50      57 recoverable_data_overflow:
862:D      -50      58 marginal_data:
863:D      -50      59 recoverable_data:
864:D      -50      60 eb60:
865:D      -50      61 maintenance_track_overflow:
866:D      -50      62 eb62:
867:D      -50      63 eb63:

```



```

866:D -50 2 $page$
867:D -50 2
868:C 2 begin (configure)
869:S 2
870:C 2 with uep^ do
871:C 3 begin
872:C 3
873:C 3 escape_caught := false;
874:C 3 saved_ureportchange := ureportchange;
875:C 3 try
876:C 4 ureportchange := false; {NEVER report media change while in configure}
877:S 4
878:C 4 (configure the control unit)
879:C 4
880:C 4 repeat
881:C 5 if set_release(uep, allow_release_timeout, disable_auto_release)<>0 then
882:C 5 handle_bad_status(uep, false, retry_required);
883:C 5 until not retry_required;
884:C 5
885:C 4 repeat
886:C 5 retry_required := false;
887:C 5 if set_status_mask(uep, my_status_mask)<>0 then
888:C 5 handle_bad_status(uep, false, retry_required);
889:C 5 until not retry_required;
890:C 5
891:C 4
892:C 4 (configure the required unit)
893:C 4
894:C 4 repeat
895:C 5 retry_required := false;
896:C 5 if chan_indep_clr(uep)<>0 then
897:C 5 handle_bad_status(uep, false, retry_required);
898:C 5 until not retry_required;
899:C 5
900:C 4 repeat
901:C 5 retry_required := false;
902:C 5 if set_unitvol(uep)<>0 then
903:C 5 handle_bad_status(uep, false, retry_required);
904:C 5 until not retry_required;
905:C 5
906:C 4 repeat
907:C 5 retry_required := false;
908:C 5 if describe(uep, describe_bytes)<>0 then
909:C 5 handle_bad_status(uep, false, retry_required);
910:C 5 until not retry_required;
911:C 5
912:C 4 with describe_bytes do
913:C 5 begin
914:C 5
915:C 5 bcd_prod_num.dn := dn;
916:C 5 prod_num := 0;
917:C 5 for index := 1 to 5 do
918:C 5 prod_num := prod_num*10+bcd_prod_num.bcd[index];
919:C 5
920:C 5
921:C 5 if ( (devid<>prod_num) and (devid<>-1) ) {wrong product number}
922:C 5 or (dt<0) {can't detect media change}
923:C 5 then ioresc_bkgnd(uep, znodevice);
924:C 5
925:C 5 dvrtemp2 := 0;

```

```

926:C 5 index := nbpb;
927:C 5 while (index>0) and not odd(index) do
928:C 6 begin
929:C 6 dvrtemp2 := dvrtemp2+1;
930:C 6 index := index div 2;
931:C 6 end; {while}
932:C 5 if index<>1 then {blocksize isn't a power of 2!}
933:C 6 dvrtemp2 := -1; {don't panic; might be just no medium present}
934:C 5
935:C 5 fixed_volume_byte.b := fvb; {fixed volume byte}
936:C 5 ulsfixed := fixed_volume_byte.bit[7-dv];
937:C 5
938:C 5 if devid=-1 then {variable-sized removeable volume; set its size}
939:C 6 umaxbytes := (maxsvadd.lfb+1)*nbpb;
940:C 5
941:C 5 if dt=2 then {it's a tape}
942:C 6 repeat {enable auto-jump sparing}
943:C 7 retry_required := false;
944:C 7 if set_options(uep, 4)<>0 then
945:C 8 handle_bad_status(uep, false, retry_required);
946:C 7 until not retry_required;
947:C 6
948:C 5 repeat
949:C 6 retry_required := false;
950:C 6 if set_status_mask(uep, my_status_mask)<>0 then
951:C 6 handle_bad_status(uep, false, retry_required);
952:C 6 until not retry_required;
953:C 5
954:C 5 end; {with}
955:C 5
956:C 4 recover
957:C 4 escape_caught := true;
958:C 4 ureportchange := saved_ureportchange;
959:C 4 if escape_caught then
960:C 4 escape:=escapecode;
961:C 4
962:C 3 end; {with}
963:C 2 end; {configure}
964:C 1 end; {CS80dsr}
965:C
966:C

```



```

967:D      1 $page$
968:S
969:D      1 module CS80dvr; {Command Set '80 Driver}
970:S
971:D      1 import
972:D      1 sysglobals, asm, mini, drvasm, bkgnd, discHPI8, tapebuf, CS80, CS80dsr;
973:S
974:D      1 export
975:D      1 type
976:D      1 mp_type = {media parameters}
977:D      1 record
978:D      1     tpm: integer; {tracks per medium}
979:D      1     bpt: integer; {bytes per track}
980:D      1 end;
981:S
982:D      1 procedure get_letter(uep: uep_type; ident: shortint; var letter: char);
983:D      1
984:D      1 procedure get_parms(var devtype: byte; var devid: integer;
985:D      2     var hardvols: shortint; var mp: mp_type);
986:S
987:D      1 procedure CS80io(fp: fibp; request: amrequesttype;
988:D      2     anyvar buffer: window; length, position: integer);
989:S
990:D      1 implement {CS80dvr}
991:S
992:D      1 var
993:D      -2 1 CS80_devtype: byte;
994:D      -6 1 CS80_devid: integer;
995:D      -8 1 CS80_hardvols: shortint;
996:D      -16 1 CS80_mp: mp_type;
997:S
998:D      1 procedure clear_unit(uep: uep_type);
999:D      2 begin {clear_unit}
1000:C      2 try
1001:C      3 allocate_bkgnd_info(uep);
1002:C      3 HPI8check_sc(uep);
1003:C      3 if HPI8amigo_identify(uep) div 256<>2 then
1004:C      4 ioresc_bkgnd(uep, znodvice);
1005:C      4 configure(uep);
1006:C      4 deallocate_bkgnd_info(uep);
1007:C      3 recover
1008:C      3 abort_bkgnd_process(uep);
1009:C      2 end; {clear_unit}
1010:C
1011:S
1012:S
1013:S      {
1014:S      -16 1 } procedures for CTA8LE self-configuration
1015:D
1016:S
1017:D      1 procedure get_parms(var devtype: byte; var devid: integer;
1018:D      2     var hardvols: shortint; var mp: mp_type);
1019:C      2 begin {get_parms}
1020:C      2 devtype := CS80_devtype;
1021:C      2 devid := CS80_devid;
1022:C      2 hardvols := CS80_hardvols;
1023:C      2 mp := CS80_mp;
1024:C      2 end; {get_parms}

```

```

1025:D      -16 1 $page$
1026:S
1027:D      1 procedure get_letter(uep: uep_type; ident: shortint; var letter: char);
1028:D      2 var
1029:D      2     retry_required: boolean;
1030:D      -38 2 describe_bytes: describe_type;
1031:D      -38 2 bcd_prod_num: {within the describe bytes}
1032:D      -38 2 packed record case integer of
1033:D      -38 2     0: (dn: usgn24);
1034:D      -38 2     1: (bcd: packed array[1..6] of usgn4);
1035:D      -42 2 end;
1036:D      -44 2 index: signed16;
1037:D      -44 2 volumes_byte: {with the describe bytes}
1038:D      -44 2 packed record case integer of
1039:D      -44 2     0: (vb: usgn8);
1040:D      -44 2     1: (bools: packed array[0..7] of boolean);
1041:D      -46 2 end;
1042:C      2 begin {get_letter}
1043:C      2 uep.ureportchange := false; {don't report media changes/power-on now!!!}
1044:S
1045:C      2 repeat {cmd w/o execution msge avoids escape if in power-on holdoff!}
1046:C      3     retry_required := false;
1047:C      3     if set_unitvol(uep)<>0 then
1048:C      4         handle_bad_status(uep, false, retry_required); {don't configure!!!}
1049:C      3     until not retry_required;
1050:S
1051:C      2 repeat
1052:C      3     retry_required := false;
1053:C      3     if describe_bytes(uep, describe_bytes)<>0 then
1054:C      4         handle_bad_status(uep, false, retry_required); {don't configure!!!}
1055:C      3     until not retry_required;
1056:S
1057:C      2 with describe_bytes do
1058:C      3     begin
1059:C      3         CS80_devtype := dt;
1060:S
1061:C      3         bcd_prod_num.dn := dn;
1062:C      3         CS80_devid := 0;
1063:C      3         for index := 1 to 5 do
1064:C      4             CS80_devid := CS80_devid*10+bcd_prod_num.bcd[index];
1065:S
1066:C      3         volumes_byte.vb := fvb+rvb;
1067:C      3         CS80_hardvols := 0;
1068:C      3         for index := 0 to 7 do
1069:C      4             CS80_hardvols := CS80_hardvols+ord(volumes_byte.bools[index]);
1070:S
1071:C      3         with CS80_mp do
1072:C      4         begin
1073:C      4             tpm := (maxcadd+1)*(maxhadd+1); {tracks per medium}
1074:C      4             if tpm=1 {only single-vector addressing info given}
1075:C      5                 then bpt := (maxsadd+1)*nbpb {bytes per track}
1076:C      5                 else bpt := (maxsadd+1)*nbpb; {bytes per track}
1077:C      4             end; {with}
1078:S
1079:C      3         end; {with}
1080:C      2     letter := 'Q';
1081:C      2 end; {get_letter}

```



```

1082:D -16 1 $page$
1083:S
1084:S {
1085:S low-level read/write routines
1086:D -16 1 }
1087:S
1088:D 1 procedure flagit(uep: anyptr);
1089:C 2 begin {flagit}
1090:C 2 bip_type(uep^).dvrtemp^.xfr_chain_semaphore := false;
1091:C 2 end; {flagit}
1092:S
1093:S
1094:D 1 procedure xfr(uep: uep_type; request: amrequesttype;
1095:D 2 bufptr: anyptr; block_address, length: integer);
1096:D 2 var
1097:D 2 command: CMD_type;
1098:D -2 2 retry_required: boolean;
1099:D -3 2 begin {xfr}
1100:C 2 allocate_bkgnd_info(uep);
1101:C 2 with bip_type(uep^).dvrtemp^ do
1102:C 3 try
1103:C 4 if HPI8amigo_identify(uep) div 256 <> 2 then
1104:C 5 ioresc_bkgnd(uep, znodevice);
1105:C 4
1106:C 4 read_operation := (request=readbytes) or (request=startread);
1107:C 4 if read_operation
1108:C 5 then command := CMDlocate_and_read
1109:C 5 else command := CMDlocate_and_wrt;
1110:C 4 ICuvalc(uep, block_address, length, command);
1111:S
1112:C 4 if length > 0 then
1113:C 5 begin
1114:C 5 HPI8wait_for_ppol(uep);
1115:C 5 xfr_chain_semaphore := true; {merely a flag for xfr busy here}
1116:C 5 HPI8upon_dxfr_comp(uep, execution_sec, bufptr, length, flagit);
1117:C 5 while xfr_chain_semaphore do
1118:C 6 {nothing};
1119:C 5 if ioresc <> inoerror then escape(-10);
1120:C 5 end {if}
1121:C 5 else
1122:C 5 bdx_pre_eoi := false;
1123:S
1124:C 4 HPI8wait_for_opol(uep);
1125:C 4 retry_required := false;
1126:C 4 if qstat(uep) <> 0 then
1127:C 5 handle_bad_status(uep, true, retry_required)
1128:C 5 else if bdx_pre_eoi then
1129:C 6 ioresc_bkgnd(uep, zcatchall); {unresolved premature eoi}
1130:S
1131:C 4 deallocate_bkgnd_info(uep);
1132:C 4 recover
1133:C 4 abort_bkgnd_process(uep);
1134:C 2 ioresult := uep^dvrtemp;
1135:C 2 if (ioresult <> ord(inoerror)) or retry_required then
1136:C 3 escape(-10);
1137:C 2 end; {xfr}

```

```

1138:D -16 1 $page$
1139:S
1140:S {
1141:S tapebuf manipulation routines
1142:D -16 1 }
1143:S
1144:D 1 procedure flush_tapebuf;
1145:D 2 var
1146:D -1 2 escape_caught: boolean;
1147:D -2 2 saved_ureportchange: boolean;
1148:C 2 begin {flush_tapebuf}
1149:C 2 if tapebuf_state=modified then
1150:C 3 with tapebuf_uep^ do
1151:C 4 begin
1152:C 4 escape_caught := false;
1153:C 4 saved_ureportchange := ureportchange;
1154:C 4 try
1155:C 5 ureportchange := true; {don't flush out to different media!}
1156:C 5 tapebuf_state := undefined; {while attempting the write}
1157:C 5 xfr(tapebuf_uep, writebytes, tapebuf_ptr, tapebuf_block, tapebuf_size);
1158:C 5 tapebuf_state := unmodified; {write was successful}
1159:C 5 recover
1160:C 5 escape_caught := true;
1161:C 4 ureportchange := saved_ureportchange;
1162:C 4 if escape_caught then
1163:C 5 escape[escapecode];
1164:C 4 end; {with}
1165:C 2 end; {flush_tapebuf}
1166:S
1167:S
1168:D 1 procedure load_tapebuf(uep: uep_type; request: amrequesttype; block: integer);
1169:D 2 var
1170:D -1 2 xfr_required: boolean;
1171:C 2 begin {load_tapebuf}
1172:C 2 xfr_required := (tapebuf_uep <> uep) or (tapebuf_block <> block) or (tapebuf_state=undefined);
1173:C 2 if xfr_required then
1174:C 3 begin
1175:C 3 flush_tapebuf;
1176:C 3 tapebuf_uep := uep;
1177:C 3 tapebuf_block := block;
1178:C 3 tapebuf_state := undefined;
1179:C 3 end; {if}
1180:S
1181:C 2 if not xfr_required {then confirm media present & unchanged}
1182:C 3 or (request=writebytes) {then confirm media not write protected}
1183:C 3 then
1184:C 4 begin
1185:C 4 xfr(tapebuf_uep, request, nil, tapebuf_block, 0);
1186:C 4 if tapebuf_state=undefined then
1187:C 5 xfr_required := true;
1188:C 4 end; {if}
1189:S
1190:C 2 if xfr_required then
1191:C 3 begin
1192:C 3 tapebuf_size := shifted_left(1, uep^dvrtemp2);
1193:C 3 xfr(tapebuf_uep, readbytes, tapebuf_ptr, tapebuf_block, tapebuf_size);
1194:C 3 tapebuf_state := unmodified; {read was successful}
1195:C 3 end; {if}
1196:C 2 end; {load_tapebuf}

```



```

1197:D -16 1 $page$
1198:S
1199:S
1200:S { read/write routine
1201:S
1202:S The new Subset/80 devices coming out which support multiple block
1203:S sizes have forced us to abandon the 2.X driver's essentially never-
1204:S used asynchronous capabilities. It's simply too difficult to handle
1205:S the media change situation when the media's block size also changes.
1206:S For instance, an asynchronous transfer started on the 256-byte block
1207:S assumption might discover new media formatted to 1024-byte blocks, in
1208:S which case entire transfer would need re-starting, this time using
1209:S tapebuf for buffering. This situation would not be detected until
1210:S well into the asynchronous transfer chain, at which point we could
1211:S be deadlocked
1212:S
1213:S -16 1 }
1214:D
1215:D 1 procedure transfer(uep: uep_type; fp: fibp; request: amrequesttype;
1216:D 2 bufptr: Charptr; abs_position, length: integer);
1217:D 2 var
1218:D -1 2 retry_required: boolean;
1219:D -4 2 blockpower: shortint;
1220:D -8 2 blocksize: integer;
1221:D -20 2 block, intra_block_offset, partial_length: integer;
1222:D
1223:D 2 begin (transfer)
1224:D 2 repeat
1225:D 3 retry_required := false;
1226:D 3
1227:D 3 try
1228:D 4 if uep^.dvrtemp2<0 then (block size unknown; try to determine)
1229:D 5 begin
1230:D 5 clear_unit(uep);
1231:D 5 ioresult := uep^.dvrtemp;
1232:D 5 if ioresult<>ord(inoerror) then
1233:D 6 escape(-10);
1234:D 5 end; (if)
1235:D 4
1236:D 4 blockpower := uep^.dvrtemp2;
1237:D 4 if blockpower<0 then
1238:D 5 ioresc(znomedium); (this or block size isn't a power of 2!!!)
1239:D 4
1240:D 4 blocksize := shifted_left(1, blockpower);
1241:D 4 if blocksize>tapebuf_maxsize then
1242:D 5 ioresc(zuninitialized); (our buffer is too small to handle)
1243:D 4
1244:D 4 block := shifted_right(abs_position, blockpower);
1245:D 4 intra_block_offset := mod_power_of_2(abs_position, blockpower);
1246:D 4
1247:D 4 if blockpower<=8 then
1248:D 5 begin (handle a 256-byte or smaller block media)
1249:D 5 if intra_block_offset<>0 then ioresc(zbadmode);
1250:D 5 xfr(uep, request, bufptr, block, length);
1251:D 5 end (handle a 256-byte or smaller block device)

```

```

1252:C 5 $page$
1253:S
1254:C 5 else
1255:C 5 begin (handle buffering for up to tapebuf_maxsize block media)
1256:C 5
1257:C 5 partial_length := blocksize-intra_block_offset;
1258:C 5 if partial_length>length then partial_length := length;
1259:C 5
1260:C 5 case request of
1261:C
1262:C 6 readbytes, startread:
1263:C 6 begin (read operations)
1264:C 6
1265:C 6 if intra_block_offset>0 then (partial block at front)
1266:C 7 begin
1267:C 7 load_tapebuf(uep, readbytes, block);
1268:C 7 moveleft(tapebuf_ptr^[intra_block_offset], bufptr^, partial_length);
1269:C 7 bufptr := addr(bufptr^, partial_length);
1270:C 7 abs_position := abs_position+partial_length;
1271:C 7 block := shifted_right(abs_position, blockpower);
1272:C 7 length := length-partial_length;
1273:C 7 end; (if)
1274:C 6
1275:C 6 if length=blocksize then (one or more blocks remain)
1276:C 7 begin
1277:C 7 flush_tapebuf; (because we may travel far, far away)
1278:C 7 xfr(uep, readbytes, bufptr, block, length);
1279:C 7 end
1280:C 6 else if length>0 then (partial block at back)
1281:C 7 begin
1282:C 7 load_tapebuf(uep, readbytes, block);
1283:C 7 moveleft(tapebuf_ptr^, bufptr^, length);
1284:C 7 end;
1285:C 6 end; (read operations)
1286:C 6
1287:C 6 writebytes, startwrite:
1288:C 6 begin (write operations)
1289:C 6
1290:C 6 if intra_block_offset>0 then (partial block at front)
1291:C 7 begin
1292:C 7 load_tapebuf(uep, writebytes, block);
1293:C 7 tapebuf_state := modified;
1294:C 7 moveleft(bufptr^, tapebuf_ptr^[intra_block_offset], partial_length);
1295:C 7 bufptr := addr(bufptr^, partial_length);
1296:C 7 abs_position := abs_position+partial_length;
1297:C 7 block := shifted_right(abs_position, blockpower);
1298:C 7 length := length-partial_length;
1299:C 7 end; (if)
1300:C 6
1301:C 6 partial_length := length-mod_power_of_2(length, blockpower);
1302:C 6 if partial_length>0 then (one or more whole blocks remain)
1303:C 7 begin
1304:C 7 flush_tapebuf; (because we may travel far, far away)
1305:C 7 tapebuf_state := undefined; (in case this overwrites!)
1306:C 7 xfr(uep, writebytes, bufptr, block, partial_length);
1307:C 7 bufptr := addr(bufptr^, partial_length);
1308:C 7 abs_position := abs_position+partial_length;
1309:C 7 block := shifted_right(abs_position, blockpower);
1310:C 7 length := length-partial_length;
1311:C 7

```



```

1312:C      7      end; (if)
1313:C
1314:C      6      if length>0 then (a partial block remains)
1315:C      7      begin
1316:C      7          load_tapebuf(uep, writebytes, block);
1317:C      7          tapebuf_state := modified;
1318:C      7          moveleft(bufptr^, tapebuf_ptr^, length);
1319:C      7      end; (if)
1320:C
1321:C      6      flush_tapebuf; (so errors get reported in the right place!)
1322:C
1323:C      6      end; (write operations)
1324:C
1325:C      6      end; (case)
1326:C
1327:C      5      end; (handle buffering for up to tapebuf_maxsize block media)
1328:C
1329:C      4      recover
1330:C      4      begin
1331:C      4          if escapecode<>-10 then
1332:C      5              escape(escapecode);
1333:C      4          if ioresult=ord(inoerror) then (media changed; restart)
1334:C      5              retry_required := true;
1335:C      4          end; (recover)
1336:C
1337:C      3      until not retry_required;
1338:C
1339:C      2      if (request=startread) or (request=startwrite) then
1340:C      3          call(fp^.feot, fp) (call the end of transfer procedure)
1341:C      3      else
1342:C      3          uep^.dvrtemp := ord(inoerror); (report synchronous errors only once)
1343:C
1344:C      2      end; (transfer)

```

```

1345:D      -16 1 $page$
1346:S
1347:C      (
1348:C      CS80 transfer method request handler
1349:D      -16 1 )
1350:C
1351:D      1 procedure CS80io;
1352:D      2 var
1353:D      -4 2 uep: uep_type;
1354:C      2 begin (CS80io)
1355:S
1356:C      2 ioresult := ord(inoerror);
1357:C      2 uep := addr(unitable^[fp^.funit]);
1358:C
1359:C      2 if uep^.offline then
1360:C      3 ioresult := ord(znodevice)
1361:C      3 else
1362:C      3 case request of
1363:S
1364:C      4 clearunit:
1365:C      4 begin
1366:C      4 clear_unit(uep);
1367:C      4 ioresult := uep^.dvrtemp;
1368:C      4 uep^.dvrtemp := ord(inoerror); (report synchronous errors only once)
1369:C      4 if uep=tapebuf_uep then tapebuf_state := undefined;
1370:C      4 end;
1371:S
1372:C      4 unitstatus:
1373:C      4 fp^.fbusy := unit_busy(uep);
1374:C
1375:C      4 flush:
1376:C      4 begin
1377:C      4 if uep=tapebuf_uep then
1378:C      5 try
1379:C      6 flush_tapebuf;
1380:C      6 recover
1381:C      6 if escapecode<>-10 then escape(escapecode);
1382:C      4 uep^.dvrtemp := ord(inoerror); (report synchronous errors only once)
1383:C      4 end;
1384:S
1385:C      4 readbytes, writebytes, startread, startwrite:
1386:C      4 begin
1387:C      4 uep^.dvrtemp := ord(inoerror); (report synchronous errors only once)
1388:C      4 if Simon_no_DMA(uep) then
1389:C      5 ioresult := ord(zbadhma)
1390:C      5 else if uep^.ureportchange and not uep^.umediavalid then
1391:C      6 ioresult := ord(zmediumchanged)
1392:C      6 else if (position<0) or (length<0) or (position+length>fp^.fpeof) then
1393:C      7 ioresult := ord(ieof)
1394:C      7 else
1395:C      7 transfer(uep, fp, request, addr(buffer), position+fp^.fileid+uep^.byteoffset, length);
1396:C      4 end;
1397:S
1398:C      4 otherwise (unrecognized request)
1399:C      4 ioresult := ord(ibadrequest);
1400:S
1401:C      4 end; (cases)
1402:C      2 end; (CS80io)
1403:S
1404:C      1 end; (CS80dvr)

```



```
1405:D      1 $page$
1406:S
1407:S
1408:D      1 { program CS80init }
1409:S
1410:D      1 import
1411:D      1   tapebuf, loader;
1412:S
1413:C      1 begin {CS80init}
1414:C      1   init_tapebuf;
1415:C      1   markuser;
1416:C      1 end. {CS80init}
1417:S
```

No errors. No warnings.

***** Nonstandard language features enabled *****

CTABLE

Description

CTABLE is the automatic-configuration program, whose code file (named TABLE) is executed at boot-up.

Notes

Refer to the Pascal 3.0 Workstation System manual for details on how to modify this program. TABLE can be re-executed at any time to re-initialize the device table.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:0
2:0
3:0
4:0
5:0
6:0
7:0
8:0
9:0
10:0
11:0
12:0
13:0
14:0
15:0
16:0
17:0
18:0
19:0
20:0
21:0

```

(c) Copyright Hewlett-Packard Company, 1983.
All rights are reserved. Copying or other
reproduction of this program except for archival
purposes is prohibited without the prior
written consent of Hewlett-Packard Company.

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government
is subject to restrictions as set forth in
paragraph (b) (3) (B) of the Rights in Technical
Data and Computer Software clause in
DAR 7-104.9(a).

HEWLETT-PACKARD COMPANY
Fort Collins, Colorado

(*)

*)

```

22:D 0 $page, sysprog$
23:S
24:D 0 (*****
25:D 0 (*)
26:D 0 (* Note: You will need to use the following *)
27:D 0 (* compiler directive if the 'INTERFACE' *)
28:D 0 (* file is not in your current LIBRARY. *)
29:D 0 (* Modify the volume name as necessary for *)
30:D 0 (* your configuration. *)
31:D 0 (*)
32:D 0 (* $search 'CONFIG:INTERFACE.'$ *)
33:D 0 (*)
34:D 0 (*****
35:S
36:D 0 program {self-configuring} ctable;
37:S
38:D 1 module options;
39:S
40:D 1 (*****
41:D 1 (* Choose the desired configuration options *)
42:D 1 (* by editing the CONSTANT declarations in *)
43:D 1 (* this module. *)
44:D 1 (*****
45:S
46:D 1 import
47:D 1 sysglobals;
48:S
49:D 1 export
50:S
51:D 1 {power-up system unit}
52:D 1 const
53:D 1 specified_system_unit =
54:D 1 0; (<> overrides auto-assignment)
55:S
56:S
57:D 1 {local mass storage directory access method}
58:D 1 type
59:D 1 lms_dam_type = {local mass storage dam}
60:D 1 (- LIF, UCSD );
61:D 1 const
62:D 1 primary_lms_dam =
63:D 1 LIF;
64:S
65:S
66:D 1 {floppy/harddisc unit number slot tradeoff's}
67:D 1 const
68:D 1 floppy_unit_pairs = {[1..10]}
69:D 1 3;
70:D 1 first_harddisc_lun = {do not edit!}
71:D 1 7*(floppy_unit_pairs-1)*2;
72:D 1 last_harddisc_lun =
73:D 1 40;
74:S

```



```

75:D 1 $page$
76:S
77:D 1 {local printer type option}
78:D 1 type
79:D 1 local_printer_type = (HP1B, RS232);
80:D 1 const
81:D 1 local_printer_option = HP1B;
82:S
83:S
84:D 1 {local printer timeout}
85:S
86:S {
87:S maximum allowed delay between any two bytes:
88:S >0 specifies milliseconds (up to one hour)
89:S =0 specifies infinite timeout
90:S
91:S recommended values:
92:S - HP2630 series (HP-1B) 3000
93:S - HP2670 series (HP-1B) 3000
94:S - HP8876 (HP-1B) 7000
95:S - HP82905 (HP-1B) 12000
96:S Note: the HP82905 is currently NOT supported
97:S due to its improper response to interface
98:S clear (IFC), and its incompatible graphics
99:S dump sequence.
100:D 1 }
101:D 1 const
102:D 1 local_printer_timeout =
103:D 1 $IF local_printer_option=HP1B$
104:D 1 12000; {milliseconds}
105:S $END$
106:S $IF local_printer_option=RS232$
107:D 1 0; {infinite}
108:S $END$
109:S
110:D 1 {default dav's for devices not found by scanning}
111:D 1 type
112:D 1 dav_type = {device address vector}
113:D 1 packed record
114:D 1 sc, ba, du, dv: -128..127;
115:D 1 end;
116:D 1 const
117:D 1 HP9885_default_dav =
118:D 1 dav_type[sc: 12, ba: -1, du: 0, dv: -1];
119:D 1 SRM_default_dav =
120:D 1 dav_type[sc: 21, ba: {node} 0,
121:D 1 du: {unit} 8, dv: -1];
122:D 1 BUBBLE_default_dav =
123:D 1 dav_type[sc: 30, ba: 0, du: 0, dv: 0];
124:D 1 local_HP1B_printer_default_dav =
125:D 1 dav_type[sc: 7, ba: 1, du: -1, dv: -1];
126:D 1 local_RS232_printer_default_dav =
127:D 1 dav_type[sc: 9, ba: 0, du: -1, dv: -1];
128:S

```

```

129:D 1 $page$
130:S
131:D 1 {local hard disc partitioning parameters}
132:D 1 type
133:D 1 pp_type = {partitioning parameters}
134:D 1 record
135:D 1 mvs: integer; {min vol size in bytes}
136:D 1 mnv: shortint; {max number of volumes}
137:D 1 end;
138:S {
139:S In general, MNV puts an upper bound on the
140:S number of logical volumes that a physical
141:S volume can be partitioned into. Depending
142:S upon MNV's range, however, several types of
143:S behavior can occur.
144:S If MNV=0, then no logical volumes will ever
145:S be assigned for the device.
146:S If abs(MNV)=1, then exactly one logical
147:S volume will be assigned per physical volume
148:S of the device. This corresponds to the
149:S 2.X CTABLE's "single volume" mode.
150:S If MNV>1, then partitioning will always be
151:S performed, subject to meeting the minimum
152:S volume size restrictions. This corresponds
153:S to the 2.X CTABLE's "multi volume" mode.
154:S If MNV<-1, then partitioning will be
155:S performed, but afterwards any logical volume
156:S that does not contain a valid directory will
157:S be coalesced with a previous adjacent logical
158:S volume if that one DOES contain a valid
159:S directory. As an extreme case, if only a
160:S single directory exists, and it is at the
161:S beginning of the physical volume, then all
162:S following logical volumes will be coalesced
163:S with the first providing the same behavior
164:S as the 2.X CTABLE's "auto volume" mode. With
165:S the less extreme cases, a wide variety of
166:S partitioning options are now possible without
167:S modification to CTABLE.
168:D 1 }
169:D 1 const
170:D 1 min_size = {in bytes [1..maxint]}
171:D 1 1000000;
172:D 1 max_vols = {[ -30..30]; <0 means autocoesce}
173:D 1 30;
174:D 1 HP913X_A_pp =
175:D 1 pp_type[mvs: min_size, mnv: max_vols];
176:D 1 HP913X_B_pp =
177:D 1 pp_type[mvs: min_size, mnv: max_vols];
178:D 1 HP913X_C_pp =
179:D 1 pp_type[mvs: min_size, mnv: max_vols];
180:D 1 CS803isc_pp =
181:D 1 pp_type[mvs: min_size, mnv: max_vols];

```



```

180:D      1 $page$
181:S
182:D      1 {system unit auto-search declarations}
183:D      1 const
184:D      1     sysunit_list_length =
185:D      1     7;
186:D      1 type
187:D      1     sysunit_list_type =
188:D      1     array[1..sysunit_list_length] of unitnum;
189:D      1 const
190:D      1     sysunit_list =
191:D      1     sysunit_list_type[
192:D      1         first_harddisc lun,
193:D      1         45, {srm, prefixed to user's sysvol}
194:D      1         4,  {floppy unit 1, primary DAM}
195:D      1         44, {floppy unit 1, secondary DAM}
196:D      1         3,  {floppy unit 0, primary DAM}
197:D      1         43, {floppy unit 0, secondary DAM}
198:D      1         42]; {bubble}
199:S
200:S
201:D      1 {HP-IB select code scanning declarations}
202:D      1 const
203:D      1     sc_list_length =
204:D      1     3;
205:D      1 type
206:D      1     sc_list_type =
207:D      1     array[1..sc_list_length] of shortint;
208:D      1 const
209:D      1     sc_list =
210:D      1     sc_list_type[
211:D      1         7, {internal HP-IB}
212:D      1         8, {default sc for HP98624 HP-IB}
213:D      1         14]; {default sc for HP98625 HP-IB}
214:S
215:S
216:D      1 implement {options}
217:S
218:C      1 end; {options}

```

```

219:D      1 $page, range off, ovflcheck off, partial_eval on$
220:S
221:D      1 module ctr; {ctable routines}
222:S
223:D      1
224:D      1     (*****
225:D      1     *                               *
226:D      1     *      Warning:                 *
227:D      1     *      This module should not be modified! *
228:D      1     *                               *
229:D      1     *****)
230:S
231:D      1 import
232:D      1     sysglobals, loader, options, ldr;
233:S
234:S
235:D      1 export
236:S
237:D      1 const {mass storage letter specifiers}
238:D      1     INTERNAL = 'M';
239:D      1     HP8290X = 'N';
240:D      1     HP9885 = 'F';
241:D      1     HP9885 = 'H';
242:D      1     HP913X_A = 'U';
243:D      1     HP913X_B = 'V';
244:D      1     HP913X_C = 'W';
245:D      1     CS80 = 'Q';
246:D      1     SRM = 'G';
247:D      1     PRINTER = 'I';
248:D      1     RAM = 'R';
249:D      1     BUBBLE = 'B';
250:D      1     EPROM = 'E';
251:D      1     NODEVICE = #255;
252:S
253:D      1 type
254:D      1     flpy_flags_type = {flags governing floppy unit pair assignments}
255:D      1     packed record
256:D      1         assign_even_unit, assign_odd_unit: boolean;
257:D      1     end;
258:S
259:D      1 const
260:D      1     assign_neither_flpy_unit =
261:D      1     flpy_flags_type[assign_even_unit: false, assign_odd_unit: false];
262:D      1     assign_both_flpy_units =
263:D      1     flpy_flags_type[assign_even_unit: true, assign_odd_unit: true];
264:S
265:D      1 type
266:D      1     MSUS_type = {Mass Storage Unit Specifier}
267:D      1     record
268:D      1         flpy_flags: flpy_flags_type;
269:D      1         letter: char; {from the above mass storage letter specifiers}
270:D      1         dav: dav_type;
271:D      1     end;

```



```

271:D      1 $page$
272:S
273:D      1      mp_type = {medium parameters}
274:D      1      record
275:D      1      { tpm: integer; {tracks per medium}
276:D      1      { bpt: integer; {bytes per track}
277:D      1      end;
278:S
279:D      1      ds_type = {Directory access method Specifier for local mass storage}
280:D      1      { primary_dam, {either LIF or UCSD, as specified in options}
281:D      1      { secondary_dam, {the one not selected as primary}
282:D      1      { LIF_dam, {LIF regardless of primary/secondary choice}
283:D      1      { UCSD_dam }; {UCSD, regardless of primary/secondary choice}
284:S
285:D      1      var
286:D      -6 1      bootdev_MSUS: MSUS_type;
287:D      -8 1      bootdev_lun: unitnum;
288:S
289:S
290:D      -8 1      procedure create_temp_unitable;
291:D      -8 1      procedure assign_and_clear_unit(lunit: unitnum);
292:D      -8 1      procedure assign_temp_unitable;
293:D      -8 1      function sysunit_ok(system_unit: unitnum): boolean;
294:D      -8 1      procedure zap_assigned_unit(lunit: unitnum);
295:D      -8 1      function on_same_medium(lun1, lun2: unitnum): boolean;
296:D      -8 1      procedure remove_extraneous_volumes(first_lun, last_lun: unitnum);
297:D      -8 1      function medium_parameters(letter: char): mp_type;
298:D      -8 1      function partitioning_parameters(letter: char): pp_type;
299:D      -8 1      function number_vols(mp: mp_type, pp: pp_type): shortint;
300:D      -8 1      function svol_bytes(letter: char): integer;
301:D      -8 1      function vol_bytes(current_vol, number_vols: shortint; mp: mp_type): integer;
302:D      -8 1      function vol_offset(current_vol, number_vols: shortint; mp: mp_type): integer;
303:D      -8 1      function block_boundaries(mp: mp_type): mp_type;
304:D      -8 1      function value(symbol: string255): integer;
305:D      -8 1      function MSUSs_match(MSUS1, MSUS2: MSUS_type): boolean;
306:S
307:S
308:S
309:D      -8 1      { table entry assignment procedures }
310:S
311:D      -8 1      procedure tea_memory_volume_dam(ds:ds_type);
312:D      -8 1      procedure tea_boot(un:unitnum);
313:D      -8 1      procedure tea_srm(un:unitnum;sc,ba,du:shortint);
314:D      -8 1      procedure tea_crt(un:unitnum);
315:D      -8 1      procedure tea_kbd(un:unitnum);
316:D      -8 1      procedure tea_local_printer(un:unitnum;sc,ba:shortint;uvid:vid;bto:integer);
317:D      -8 1      procedure tea_mini(un:unitnum;ds:ds_type;du:shortint);
318:D      -8 1      procedure tea_HP885(un:unitnum;ds:ds_type;sc,du,block_os:shortint);
319:D      -8 1      procedure tea_HP895(un:unitnum;ds:ds_type;sc,ba,du,block_os:shortint);
320:D      -8 1      procedure tea_HP890X(un:unitnum;ds:ds_type;sc,ba,du:shortint);
321:D      -8 1      procedure tea_f1py(un:unitnum;lr:char;ds:ds_type;sc,ba,du:shortint);
322:D      -8 1      procedure tea_amigo_sv(un:unitnum;ds:ds_type;sc,ba,du:shortint;os:integer;lr:char;mb:integer);
323:D      -8 1      procedure tea_CS80_mv(un:unitnum;ds:ds_type;sc,ba,du,dv:shortint;os,id,mb:integer);
324:D      -8 1      procedure tea_CS80_sv(un:unitnum;ds:ds_type;sc,ba,du,dv:shortint);
325:D      -8 1      procedure tea_BUBBLE(un:unitnum;ds:ds_type;sc:shortint);
326:D      -8 1      procedure tea_EPROM(un:unitnum;ds:ds_type;sn:shortint);

```

```

327:D      -8 1 $page$
328:S
329:D      -8 1 implement {ctr}
330:S
331:D      -8 1 const {abbreviation for tea procedure calls}
332:D      -8 1      T = true;
333:D      -8 1      F = false;
334:S
335:D      -8 1 const {actual driver entry point names}
336:D      -8 1      NO_DAM_name = 'INITUNITS_NODAM';
337:D      -8 1      BOOT_DAM_name = 'BOOTDAMMODULE_BOOTDAM';
338:D      -8 1      LIF_DAM_name = 'LIFMODULE_LIFDAM';
339:D      -8 1      UCSD_DAM_name = 'UCSDMODULE_UCSD_DAM';
340:D      -8 1      UNBLOCKED_DAM_name = 'MISC_UNBLOCKEDDAM';
341:D      -8 1      SRM_DAM_name = 'SRMDAMMODULE_SRM_DAM';
342:S
343:D      -8 1      NULL_TM_name = 'INITUNITS_NOUNIT';
344:D      -8 1      BOOT_TM_name = 'BOOTDAMMODULE_BOOTTM';
345:D      -8 1      CRT_TM_name = 'SYSDEVS_CRTIO';
346:D      -8 1      KBD_TM_name = 'SYSDEVS_KBDIO';
347:D      -8 1      MINI_TM_name = 'MINI_MINIO';
348:D      -8 1      SRM_TM_name = 'SRMDAMMODULE_SRMAM';
349:D      -8 1      PRINTER_TM_name = 'PRIDVR_PRIO';
350:D      -8 1      F885_TM_name = 'F885DVR_F885IO';
351:D      -8 1      AMIGO_TM_name = 'AMIGODVR_AMIGOIO';
352:D      -8 1      CS80_TM_name = 'CS80DVR_CS80IO';
353:D      -8 1      BUBBLE_TM_name = 'BUB8LE_BUB_TM';
354:D      -8 1      EPROM_TM_name = 'EPROMS_EPROMTM';
355:S
356:S
357:D      -8 1 var
358:D      -12 1      temp_unitable: unitableptr;
359:S
360:S
361:D      1      procedure check(parameter, lower_bound, upper_bound: integer);
362:D      1      begin
363:D      1      if (parameter<lower_bound) or (parameter>upper_bound) then
364:D      1      halt(-8) {value range error}
365:D      1      end;
366:S

```



```

367 D -12 1 ipage$
368 S
369 D -256 1 function value(symbol: string255): integer;
370 D -256 2 var
371 D -260 2 modp: moddescptr;
372 D -268 2 ptr, valueptr: address;
373 D -269 2 found: boolean;
374 C 2 begin (value)
375 C 2 value := 0;
376 C 2 found := false;
377 C 2 modp := sysdefs;
378 C 2 while (modp<>nil) and not found do
379 C 3 with modp^ do
380 C 4 begin
381 C 4 ptr := defaddr;
382 C 4 while (ptr.a<defaddr.a+defsize) and not found do
383 C 5 begin
384 C 5 found := ptr.syp=symbol;
385 C 5 ptr.a := ptr.a+strlen(ptr.syp)+1;
386 C 5 ptr.a := ptr.a+ord(odd(ptr.a));
387 C 5 valueptr.a := ptr.a+2;
388 C 5 if found then
389 C 6 value := valueptr.vcp.value;
390 C 5 ptr.a := ptr.a+ptr.gvp.short;
391 C 5 end; (while)
392 C 4 modp := link;
393 C 4 end; (with modp^);
394 C 2 end; (value)
395 S
396 S
397 D -12 1 function MSUSs_match(MSUS1, MSUS2: MSUS_type): boolean;
398 D 2 begin (MSUSs_match)
399 C 2 MSUSs_match := (MSUS1.letter = MSUS2.letter) and
400 C 2 (MSUS1.dav.sc = MSUS2.dav.sc) and
401 C 2 (MSUS1.dav.ba = MSUS2.dav.ba) and
402 C 2 (MSUS1.dav.du = MSUS2.dav.du) and
403 C 2 (MSUS1.dav.dv = MSUS2.dav.dv);
404 C 2 end; (MSUSs_match)

```

```

405 D -12 1 ipage$
406 S
407 D 1 procedure tea (lowest-level Table Entry Assignment procedure)
408 D 2 (
409 D 2 un:unitnum; (unit number)
410 D 2 dam_name: string255; (directory access method)
411 D 2 tm_name: string255; (transfer method (driver))
412 D 2 p_sc: shortint; (select code)
413 D 2 p_ba: shortint; (bus address)
414 D 2 p_du: shortint; (disc unit)
415 D 2 p_dv: shortint; (disc volume)
416 D 2 p_byteoffset: integer; (physical starting byte of volume)
417 D 2 p_devid: integer; (device identifier (driver dependent))
418 D 2 p_vuid: vid; (volume id)
419 D 2 { p_drvtemp: integer; (driver temp)
420 D 2 { p_drvtemp2: shortint; (second driver temp)
421 D 2 { p_letter: char; (device specifier letter)
422 D 2 { p_offline: boolean; (unit offline flag)
423 D 2 { p_uisinteractive: boolean; (device echos input)
424 D 2 { p_umediavalid: boolean; (open files are valid)
425 D 2 { p_uuppercase: boolean; (volume name should be uppercased)
426 D 2 { p_ufixed: boolean; (medium not removable flag)
427 D 2 { p_ueportchange: boolean; (driver directive to report/ignore medium changes)
428 D 2 { p_pad: 0..1 (not used)
429 D 2 { p_uisblkd: boolean; (blocked volume flag)
430 D 2 { p_umaxbytes: integer; (volume size in bytes)
431 S -530 2 var
432 D -530 2 dam_proc:
433 D -530 2 packed record case integer of
434 D -530 2 0: (dam: damtype);
435 D -530 2 1: (value, slink: integer);
436 D -538 2 end;
437 S
438 D -538 2 tm_proc:
439 D -538 2 packed record case integer of
440 D -538 2 0: (tm: amtype);
441 D -538 2 1: (value, slink: integer);
442 D -546 2 end;
443 S
444 D 2 begin (tea)
445 D 2 if temp_unitable=nil then halt(-3); (unassigned pointer)
446 S
447 D 2 dam_proc.value := value(dam_name);
448 D 2 dam_proc.slink := 0;
449 S
450 D 2 tm_proc.value := value(tm_name);
451 D 2 tm_proc.slink := 0;
452 S
453 D 2 if (dam_proc.value<>0) and (tm_proc.value<>0) then (assign the entry)
454 D 3 begin
455 S
456 D 3 with temp_unitable^[un] do
457 D 4 begin
458 D 4 dam := dam_proc.dam;
459 D 4 tm := tm_proc.tm;
460 D 4 sc := p_sc;
461 D 4 ba := p_ba;
462 D 4 du := p_du;
463 D 4 dv := p_dv;
464 D 4 byteoffset := p_byteoffset;

```



```

465:C      4      devid      := p.devid;
466:C      4      uvid      := p.uvid;
467:C      4      dvrtmp     := 0;           {always initially zero!}
468:C      4      dvrtmp2    := -1;          {always initially -1!}
469:C      4      letter     := p.letter;
470:C      4      offline    := false;      {always initially online!}
471:C      4      uisinteractive := p.uisinteractive;
472:C      4      uediavalid  := false;      {never valid to start with}
473:C      4      uuppercase  := not p.uisblkd; {assume case is significant}
474:C      4      uisfixed    := p.uisfixed;
475:C      4      ureportchange := true;      {do report media changes}
476:C      4      pad        := 0;           {not used}
477:C      4      uisblkd     := p.uisblkd;
478:C      4      if uisblkd then
479:C      5          umaxbytes := p.umaxbytes;
480:C      4      end; {with}
481:S
482:C      3      with bootdev_MSUS, dav do {see if this entry points to it}
483:C      4          if (p.letter=letter) and {wish we could use MSUSs_match function}
484:C      5              (p.sc=sc) and (p.ba=ba) and (p.du=du) and (p.dv=dv) and
485:C      5              ((dam_name=LIF_DAM_name) or (p.letter=SRH)) and
486:C      5              (p.byteoffset=0) then {remember this unit number!}
487:C      5          bootdev_lun := un;
488:S
489:C      3      end; {if}
490:C      2      end; {tea}
491:S
492:C
493:D
494:C      1 function dam(ds: ds_type): string255;
495:C      2 begin
496:C      3     case ds of
497:C      4         primary_dam:
498:C      5             if primary_lms_dam=LIF
499:C      6                 then dam := LIF_DAM_name;
500:C      6             else dam := UCSD_DAM_name;
501:C      5         secondary_dam:
502:C      6             if primary_lms_dam=LIF
503:C      7                 then dam := UCSD_DAM_name;
504:C      7             else dam := LIF_DAM_name;
505:C      4         LIF_dam:
506:C      5             dam := LIF_DAM_name;
507:C      4         UCSD_dam:
508:C      5             dam := UCSD_DAM_name;
509:C      3     end; {case}
510:C      2 end;

```

```

510:D      -12 1 $pages
511:S
512:D      1 function medium_parameters(letter: char): mp_type;
513:D      2 const {LOGICAL sizes unless otherwise noted}
514:D      2     INTERNAL_mp = mp_type[tpm: 2* 33, bpt: 16*256];
515:D      2     HP8290X_mp   = mp_type[tpm: 2* 33, bpt: 16*256];
516:D      2     HP9885_mp    = mp_type[tpm: 1* 77, bpt: 30*256]; {physical size}
517:D      2     HP9895_mp    = mp_type[tpm: 2* 77, bpt: 30*256]; {physical size}
518:D      2     HP913X_A_mp  = mp_type[tpm: 4*152, bpt: 31*256];
519:D      2     HP913X_B_mp  = mp_type[tpm: 4*305, bpt: 31*256];
520:D      2     HP913X_C_mp  = mp_type[tpm: 6*305, bpt: 31*256];
521:D      2     BUBBLE_mp    = mp_type[tpm: 1*512, bpt: 1*256]; {1 megabit unit}
522:D      2     (BUBBLE_mp   = mp_type[tpm: 4*512, bpt: 1*256];) {4 megabit unit}
523:D      2     null_mp     = mp_type[tpm: 0, bpt: 0];
524:D      2 begin
525:D      3     case letter of
526:D      4         INTERNAL: medium_parameters := INTERNAL_mp;
527:D      4         HP8290X:  medium_parameters := HP8290X_mp;
528:D      4         HP9885:   medium_parameters := HP9885_mp;
529:D      4         HP9895:   medium_parameters := HP9895_mp;
530:D      4         HP913X_A: medium_parameters := HP913X_A_mp;
531:D      4         HP913X_B: medium_parameters := HP913X_B_mp;
532:D      4         HP913X_C: medium_parameters := HP913X_C_mp;
533:D      4         BUBBLE:  medium_parameters := BUBBLE_mp;
534:D      4         otherwise medium_parameters := null_mp;
535:D      3     end; {case}
536:D      2 end;
537:S
538:S
539:D      1 function partitioning_parameters(letter: char): pp_type;
540:D      2 const
541:D      2     null_pp = pp_type[mvs: 0, mnv: 0];
542:D      2 begin
543:D      3     case letter of
544:D      4         HP913X_A: partitioning_parameters := HP913X_A_pp;
545:D      4         HP913X_B: partitioning_parameters := HP913X_B_pp;
546:D      4         HP913X_C: partitioning_parameters := HP913X_C_pp;
547:D      4         CS80:   partitioning_parameters := CS80disc_pp;
548:D      4         otherwise partitioning_parameters := null_pp;
549:D      3     end; {case}
550:D      2 end;

```



```

659:D -12 1 $page$
660:S
661:D 1 procedure tea_HP988S(un:unitnum;ds:ds_type;sc,du,block_os:shortint);
662:D 2 var
663:D 2 os: integer;
664:D -4 2 begin
665:C 2 check(sc, 8, 31);
666:C 2 check(du, 0, 3);
667:C 2 os := block_os*512;
668:C 2 check(os, 0, svol_bytes(HP988S)-1);
669:C 2 tea(un,dam(ds),F988:_TM_name,sc,C,du,0,os,0,'',HP988S,F,F,T,svol_bytes(HP988S)-os);
670:C 2 end;
671:S
672:S
673:D 1 procedure tea_HP989S(un:unitnum;ds:ds_type;sc,ba,du,block_os:shortint);
674:D 2 var
675:D 2 os: integer;
676:D -4 2 begin
677:C 2 check(sc, 7, 31);
678:C 2 check(ba, 0, 7);
679:C 2 check(du, 0, 3);
680:C 2 os := block_os*512;
681:C 2 check(os, 0, svol_bytes(HP989S)-1);
682:C 2 tea(un,dam(ds),AMIGO_TM_name,sc,Ca,du,0,os,0,'',HP989S,F,F,T,svol_bytes(HP989S)-os);
683:C 2 end;
684:S
685:S
686:D 1 procedure tea_HP8290X(un:unitnum;ds:os_type;sc,ba,du:shortint);
687:D 2 begin
688:C 2 check(sc, 7, 31);
689:C 2 check(ba, 0, 7);
690:C 2 check(du, 0, 3);
691:C 2 tea(un,dam(ds),AMIGO_TM_name,sc,ba,du,0,0,0,'',HP8290X,F,F,T,svol_bytes(HP8290X));
692:C 2 end;
693:S
694:S
695:D 1 procedure tea_amigo_sv(un:unitnum;ds:ds_type;sc,ba,du:shortint;os:integer;lr:char;mb:integer);
696:D 2 var
697:D -4 2 medium_size: integer;
698:C 2 begin
699:C 2 check(sc, 7, 31);
700:C 2 check(ba, 0, 7);
701:C 2 check(du, 0, 7);
702:C 2 if not (lr in [HP913X_A, HP913X_B, HP913X_C]) then
703:C 2 halt(-8); {value range error}
704:C 2 medium_size := svol_bytes(lr);
705:C 2 check(os, 0, medium_size-1);
706:C 2 if os mod 256 <> 0 then halt(-8) {value range error};
707:C 2 check(mb, 1, medium_size-os);
708:C 2 tea(un,dam(ds),AMIGO_TM_name,sc,ba,du,0,os,0,'',lr,F,T,mb);
709:C 2 end;

```

```

710:D -12 1 $page$
711:S
712:D 1 procedure tea_CS80_mv(un:unitnum;ds:ds_type;sc,ba,du,dv:shortint;os,id,mb:integer);
713:S {
714:S CS80 multiple (logical) volume assignment procedure:
715:S 1) devid must match the actual HP product number as found in describe
716:S 2) offset and umaxbytes are fixed
717:S 3) the unfixed field is assigned by the driver in the clearunit procedure
718:S }
719:C 2 begin
720:C 2 check(sc, 7, 31);
721:C 2 check(ba, 0, 7);
722:C 2 check(du, 0, 14);
723:C 2 check(dv, 0, 7);
724:C 2 tea(un,dam(ds),CS80_TM_name,sc,ba,du,dv,os,id,'',CS80,F,F,T,mb);
725:C 2 end;
726:S
727:S
728:D 1 procedure tea_CS80_sv(un:unitnum;ds:ds_type;sc,ba,du,dv:shortint);
729:S {
730:S CS80 single (logical) volume assignment procedure:
731:S 1) byteoffset always assumed to be zero
732:S 2) umaxbytes is dependent upon the media loaded, thus it is set by the
733:S driver at clearunit time and whenever it detects a media change.
734:S BOTTOM LINE: the medium CANNOT be partitioned into multiple volumes!
735:S 3) the unfixed field is assigned by the driver in the clearunit procedure
736:S 4) device can either be a disc (presumably a floppy) or a tape
737:S }
738:C 2 begin
739:C 2 check(sc, 7, 31);
740:C 2 check(ba, 0, 7);
741:C 2 check(du, 0, 14);
742:C 2 check(dv, 0, 7);
743:C 2 tea(un,dam(ds),CS80_TM_name,sc,ba,du,dv,0,-1,'',CS80,F,F,T,0);
744:C 2 end;
745:S
746:S
747:D 1 procedure tea_flpj(un:unitnum;lr:char;ds:ds_type;sc,ba,du:shortint);
748:D 2 begin
749:C 2 case lr of
750:C 2 INTERNAL: tea_mini(un,ds,du);
751:C 2 HP8290X: tea_HP8290X(un,ds,sc,ba,du);
752:C 2 CS80: tea_CS80_sv(un,ds,sc,ba,du,0);
753:C 2 HP988S: tea_HP988S(un,ds,sc,du,0);
754:C 2 HP989S: tea_HP989S(un,ds,sc,ba,du,0);
755:C 2 otherwise halt(-8) {value range error}
756:C 2 end; {case}
757:C 2 end;
758:S

```



```

759 D      -12  1  $page$
760 S
761 D      1  procedure tea_BUBBLE(un:unitnum;ds:ds_type;sc:shortint);
762 S      begin
763 D          { NOTE THAT UMAXBYTES IS ASSIGNED AT CLEARUNIT TIME }
764 S          check(sc,7,31);
765 D          tea(un,dam(ds),BUBBLE_TM_name,sc,0,0,0,0,0,'',BUBBLE,F,T,T,0);
766 S      end;
767 S
768 D      1  procedure tea_EPROM(un:unitnum;ds:ds_type;sn:shortint);
769 S      begin
770 D          { NOTE THAT UMAXBYTES IS ASSIGNED AT CLEARUNIT TIME }
771 S          check(sn,0,MAXUNIT);
772 D          tea(un,dam(ds),EPROM_TM_name,0,0,0,sn,0,0,'',EPROM,F,T,T,0);
773 S      end;
774 S

```

```

775:D -12 1 $page$
776:S
777:D 1 procedure create_temp_unitable;
778:D 2 var
779:D -2 2 lunit: unitnum;
780:D 2 begin
781:C 2 2 new(temp_unitable);
782:C 2 2 tea_nunit(0); {assign one dummy entry}
783:C 2 2 for lunit := 0 to maxunit do {copy others; avoid symbol table search each time!}
784:C 2 2 temp_unitable^lunit := temp_unitable^0;
785:C 2 end;
786:S
787:S
788:D 1 procedure assign_and_clear_unit(lunit: unitnum);
789:D 2 var
790:D -662 2 f: fib;
791:D 2 begin
792:C 2 2 if temp_unitable=nil then halt(-3); {unassigned pointer}
793:C 2 2 with unitable^lunit, f do
794:C 2 2 if (letter<>'RAM') or (lunit=0) then
795:C 2 2 begin
796:C 2 2 unitable^lunit := temp_unitable^lunit;
797:C 2 2 funit := lunit;
798:C 2 2 call(tm, addr(f), clearunit, lunit, 0, 0);
799:C 2 2 offline := uisblkd and (ioresult<>0);
800:C 2 2 end;
801:C 2 end; {with}
802:S
803:S
804:D 1 procedure assign_temp_unitable;
805:D 2 var
806:D -2 2 lunit: unitnum;
807:C 2 2 begin
808:C 2 2 if temp_unitable=nil then halt(-3); {unassigned pointer}
809:C 2 2 lockfiles; {close all standard system files}
810:C 2 2 for lunit := 0 to maxunit do
811:C 2 2 assign_and_clear_unit(lunit);
812:C 2 end;
813:S
814:S
815:D 1 function sysunit_ok(system_unit: unitnum): boolean;
816:D 2 begin
817:C 2 2 sysunit := system_unit;
818:C 2 2 initsysunit;
819:C 2 2 with unitable^system_unit do
820:C 2 2 sysunit_ok := uisblkd and not offline and (uvid<>'');
821:C 2 end;
822:S
823:S
824:D 1 procedure zap_assigned_unit(lunit: unitnum);
825:D 2 begin
826:C 2 2 tea_nunit(lunit); {zap the temp unitable entry}
827:C 2 2 unitable^lunit := temp_unitable^lunit; {now zap the real one!}
828:C 2 end;

```



```

829:D -12 1 $pages$
830:S
831:D 1 function on_same_medium(lun1, lun2: unitnum): boolean;
832:D 2 var
833:D 3 uep: ^unitentry;
834:D 4 begin (on_same_medium)
835:C 2 uep := addr[untable^[lun2]];
836:C 2 with untable^[lun1] do
837:C 3 on_same_medium := (sc=uep^.sc) and (ba=uep^.ba) and
838:C 3 (du=uep^.du) and (dv=uep^.dv) and
839:C 3 (letter=uep^.letter);
840:C 2 end; (on_same_medium)
841:S
842:S
843:D 1 procedure remove_extraneous_volumes(first_lun, last_lun: unitnum);
844:D 2 var
845:D 3 first_lun_ok: boolean;
846:D 4 lun: unitnum;
847:C -1 2 begin (remove_extraneous_volumes)
848:C -4 2 first_lun_ok := false;
849:C 2 while first_lun<last_lun do
850:C 3 if first_lun_ok then
851:C 4 begin
852:C 5 lun := first_lun+1;
853:C 5 with untable^[first_lun] do
854:C 6 while (lun<=last_lun) and not sysunit_ok(lun) do
855:C 7 begin
856:C 8 if untable^[lun].byteoffset = byteoffset+umaxbytes then
857:C 9 begin
858:C 10 umaxbytes := umaxbytes+untable^[lun].umaxbytes;
859:C 11 zap_assigned_unit(lun);
860:C 12 end;
861:C 13 lun := lun+1;
862:C 14 end; (while)
863:C 15 first_lun := lun;
864:C 16 end (then)
865:C 17 else if sysunit_ok(first_lun) then
866:C 18 first_lun_ok := true
867:C 19 else
868:C 20 first_lun := first_lun+1;
869:C 21 end; (remove_extraneous_volumes)
870:S
871:C 1 end; (ctr)

```

```

872:D 1 $pages$
873:S
874:D 1 module BRstuff; {BOOTROM stuff}
875:S
876:D 1 {*****}
877:D 1 { * Warning: * }
878:D 1 { * This module should not be modified! * }
879:D 1 { * }
880:D 1 {*****}
881:S
882:S
883:D 1 import
884:D 1 sysglobals, options, ctr;
885:S
886:D 1 export
887:D 1 const
888:D 1 INTERNAL_MSUS = MSUS_type
889:D 1 [ flpy_flags: assign_neither_flpy_unit, letter: INTERNAL,
890:D 1 dav: dav_type[sc: -1, ba: -1, du: 0, dv: -1] ];
891:S
892:D 1 function internal_mini_present: boolean;
893:D 1 procedure get_bootdevice_MSUS(var MSUS: MSUS_type);
894:S
895:D 1 implement {BRstuff}
896:S
897:D 1 type
898:D 1 signed4 = -8..7;
899:D 1 signed8 = -128..127;
900:S
901:D 1 fmt_type = {format field in the msus byte}
902:D 1 (f0,f1,f2,f3,f4,f5,f6,f7);
903:S
904:D 1 dev_type = {device field in the msus byte}
905:D 1 (d0,d1,d2,d3,d4,d5,d6,d7,d8,d9,d10,d11,d12,d13,d14,d15,
906:D 1 d16,d17,d18,d19,d20,d21,d22,d23,d24,d25,d26,d27,d28,d29,d30,d31);
907:S
908:D 1 BR_msus_type = {BOOTROM's mass storage unit specifier}
909:D 1 packed record case boolean of
910:D 1 false: (8-bit unit number)
911:D 1 ( fmt: fmt_type; {directory format}
912:D 1 dev: dev_type; {device}
913:D 1 un: signed8; {8-bit unit number}
914:D 1 sc: signed8; {select code}
915:D 1 ba: signed8; {bus address} );
916:D 1 true: (4-bit volume / 4-bit unit number for CS80 discs)
917:D 1 ( pad: signed8; {format/device byte}
918:D 1 vn4: signed4; {4-bit volume number}
919:D 1 un4: signed4; {4-bit unit number} );
920:D 1 end; {BR_msus_type}
921:S
922:D 1 var
923:D 1 ROM_ID[16382]: {BOOTROM identification word}
924:D 1 shortint;
925:S
926:D 1 ncrives[-296]: {Maximum Unit for Internal Mini-Floppy}
927:D 1 packed record b: signed8; end;
928:S
929:D 1 default_msus[-292]: {boot device's msus}
930:D 1 BR_msus_type;

```



```

931:D      1 $page$
932:S
933:D      1 function internal_mini_present: boolean;
934:C      2 begin
935:C      3   if ROM_ID<0
936:C      4   then internal_mini_present := true      {1.0 BOOTROM on 9826}
937:C      5   else internal_mini_present := ndrives.b<>-1; {2.0 or greater BOOTROM}
938:C      2 end;
939:S
940:S
941:D      1 procedure get_bootdevice_MSUS(var MSUS: MSUS_type);
942:C      2 type
943:D      3 letter_table_type = array[dev_type] of char;
944:D      2 const
945:D      3 letter_table = (BOOTROM dev to Pascal letter conversion table)
946:D      4 letter_table_type
947:D      5 [ INTERNAL, NODEVICE, NODEVICE, NODEVICE, HP9895, HP8290X, HP9885, HP913X_A,
948:D      6 HP913X_B, HP913X_C, NODEVICE, NODEVICE, NODEVICE, NODEVICE, NODEVICE, NODEVICE,
949:D      7 CS80, CS80, NODEVICE, NODEVICE, NODEVICE, NODEVICE, BUBBLE, NODEVICE,
950:D      8 NODEVICE, NODEVICE, NODEVICE, NODEVICE, NODEVICE, NODEVICE, NODEVICE ];
951:C      2 begin
952:C      3   if ROM_ID<0 then {1.0 Boot ROM on 9826; internal minifloppy only}
953:C      4   MSUS := INTERNAL_MSUS
954:C      5   else {2.0 or greater Boot ROM}
955:C      6   with default_msus do
956:C      7   begin
957:C      8     if fmt=f7 then {non sector-oriented device}
958:C      9     if dev=d1
959:C      10     then MSUS.letter := SRM
960:C      11     else MSUS.letter := NODEVICE
961:C      12     else {sector-oriented device}
962:C      13     MSUS.letter := letter_table[dev];
963:C      14     MSUS.dav.sc := sc;
964:C      15     MSUS.dav.ba := ba;
965:C      16     if MSUS.letter=CS80 then
966:C      17     begin
967:C      18       MSUS.dav.du := un4;
968:C      19       MSUS.dav.dv := vn4;
969:C      20     end {then}
970:C      21     else
971:C      22     begin
972:C      23       MSUS.dav.du := un;
973:C      24       MSUS.dav.dv := 0;
974:C      25     end; {else}
975:C      26   end; {with}
976:C      3   end;
977:S
978:C      1 end; {BRstuff}

```

```

979:D      1 $page$
980:S
981:D      1 module scanstuff;
982:S
983:D      1 {*****}
984:D      1 {* Warning: *}
985:D      1 {* This module should not be modified! *}
986:D      1 {* *}
987:D      1 {*****}
988:D      1
989:S
990:D      1 import
991:D      1 sysglobals, options, ctr;
992:S
993:D      1 export
994:S
995:D      1 procedure init_scanstuff;
996:D      1 function scanneddevice_letter(scan_dav: dav_type): char;
997:D      1 procedure get_CS80_parms(CS80dav: dav_type;
998:D      2 var CS80dt: byte; var CS80id: integer;
999:D      2 var CS80hardvols: shortint; var CS80mp: mp_type);
1000:S
1001:D      1 implement {scanstuff}
1002:S
1003:S
1004:D      1 type
1005:D      1 uep_type = ^unitentry;
1006:S
1007:D      1 uep_proc_type = procedure(uep: uep_type);
1008:D      1 HP1Bget_amigo_ident_type = procedure(uep: uep_type; var ident: shortint);
1009:D      1 get_letter_type = procedure(uep: uep_type; ident: shortint; var letter: char);
1010:D      1 get_CS80_parms_type = procedure(var CS80dt: byte;
1011:D      1 var CS80id: integer;
1012:D      1 var CS80hardvols: shortint;
1013:D      1 var CS80mp: mp_type);
1014:S
1015:D      1 proc_type =
1016:D      1 packed record case integer of
1017:D      2 0: (value, slink: integer);
1018:D      3 1: (up: uep_proc_type);
1019:D      4 2: (gai: HP1Bget_amigo_ident_type);
1020:D      5 3: (gl: get_letter_type);
1021:D      6 4: (gcp: get_CS80_parms_type);
1022:D      7 end;
1023:S
1024:S
1025:D      1 var
1026:D      2 -8 1 allocate_bkgnd_info_proc: proc_type;
1027:D      3 -16 1 deallocate_bkgnd_info_proc: proc_type;
1028:D      4 -24 1 abort_bkgnd_process_proc: proc_type;
1029:D      5 -32 1 HP1Bcheck_sc_proc: proc_type;
1030:D      6 -40 1 HP1Bget_amigo_ident_proc: proc_type;
1031:S
1032:D      2 -48 1 get_amigo_letter_proc: proc_type;
1033:D      3 -56 1 get_CS80_letter_proc: proc_type;
1034:D      4 -64 1 get_CS80_parms_proc: proc_type;
1035:S
1036:D      2 -65 1 bkgnd_and_dischpib_present: boolean;

```



```

1037:D  -65 1 $page$
1038:S
1039:D  1 function scanneddevice_letter(scan_dav: dav_type): char;
1040:S
1041:D  2 type
1042:D  2   amigo_class_type = (upper three bits of the first ident byte)
1043:D  2   (storage, display, data_communication, processor,
1044:D  2   stimulus, measurement, unassigned6, unassigned7);
1045:S
1046:D  2 var
1047:D  2   ue: unitentry;
1048:D  2   ident:
1049:D  2   packed record case integer of
1050:D  2     0: (word: shortint);
1051:D  2     1: (upper_byte, lower_byte: byte);
1052:D  2     2: (amigo_class: amigo_class_type);
1053:D  2   end;
1054:S
1055:D  -8 2 procedure set_scanneddevice_letter(get_letter_proc: proc_type);
1056:D  -8 2
1057:D  -9 3 var
1058:D  -9 3   device_letter: char;
1059:C  3 begin (set_scanneddevice_letter)
1060:C  3   if get_letter_proc.value<>0 then
1061:C  4   begin
1062:C  4     call(get_letter_proc.gl, addr(ue), ident.word, device_letter);
1063:C  4     scanneddevice_letter := device_letter;
1064:C  4   end; (if)
1065:S
1066:C  3 end; (set_scanneddevice_letter)
1067:D  2 begin (scanneddevice_letter)
1068:D  2   scanneddevice_letter := NODEVICE; (until proven otherwise)
1069:D  2   if bkgnd_and_dischpib_present then
1070:C  3   try
1071:C  4   ue.sc := scan_dav.sc;
1072:C  4   ue.ba := scan_dav.ba;
1073:C  4   ue.du := scan_dav.du;
1074:C  4   ue.dv := scan_dav.dv;
1075:C  4   call(allocate_bkgnd_info_proc.up, addr(ue));
1076:C  4   call(HPIBcheck_sc_proc.up, addr(ue));
1077:C  4   call(HPIBget_amigo_ident_proc.gai, addr(ue), ident.word);
1078:C  4   if ident.amigo_class=storage then
1079:C  5   if ident.upper_byte=2
1080:C  6   then set_scanneddevice_letter(get_CS80_letter_proc)
1081:C  6   else set_scanneddevice_letter(get_amigo_letter_proc)
1082:C  6   else if ident.amigo_class=display then
1083:C  4   scanneddevice_letter := PRINTER;
1084:C  4   call(deallocate_bkgnd_info_proc.up, addr(ue));
1085:C  4   recover
1086:C  4   call(abort_bkgnd_process_proc.up, addr(ue));
1087:C  2 end; (scanneddevice_letter)

```

```

1087:D  -65 1 $page$
1088:S
1089:D  1 procedure get_CS80_parms(CS80dav: dav_type;
1090:D  2   var CS80dt: byte; var CS80id: integer;
1091:D  2   var CS80hardvols: shortint; var CS80mp: mp_type);
1092:D  2 begin (get_CS80_parms)
1093:D  2   if (scanneddevice_letter(CS80dav)=CS80) and (get_CS80_parms_proc.value<>0) then
1094:C  3   call(get_CS80_parms_proc.gcp, CS80dt, CS80id, CS80hardvols, CS80mp)
1095:C  3   else
1096:C  3   begin
1097:C  3     CS80dt := 255;
1098:C  3     CS80id := 0;
1099:C  3     CS80hardvols := 0;
1100:C  3     CS80mp := medium_parameters(NODEVICE);
1101:C  3   end; (else)
1102:C  2 end; (get_CS80_parms)
1103:S
1104:S
1105:D  1 procedure init_scanstuff;
1106:S
1107:D  2 {
1108:D  2   NOTE: all procedure variables are GLOBAL, so their static links are
1109:D  2   guaranteed to have been cleared @ load time
1110:D  2 }
1111:D  2 begin (init_scanstuff)
1112:D  2   allocate_bkgnd_info_proc.value := value('BKGND_ALLOCATE_BKGND_INFO');
1113:D  2   deallocate_bkgnd_info_proc.value := value('BKGND_DEALLOCATE_BKGND_INFO');
1114:D  2   abort_bkgnd_process_proc.value := value('BKGND_ABORT_BKGND_PROCESS');
1115:D  2   HPIBcheck_sc_proc.value := value('DISCHPIB_HPIBCHECK_SC');
1116:D  2   HPIBget_amigo_ident_proc.value := value('DISCHPIB_HPIBGET_AMIGO_IDENT');
1117:D  2   get_amigo_letter_proc.value := value('AMIGODVR_GET_LETTER');
1118:D  2   get_CS80_letter_proc.value := value('CS80DVR_GET_LETTER');
1119:D  2   get_CS80_parms_proc.value := value('CS80DVR_GET_PARMS');
1120:S
1121:D  2   bkgnd_and_dischpib_present := (allocate_bkgnd_info_proc.value<>0) and
1122:D  2   (deallocate_bkgnd_info_proc.value<>0) and
1123:D  2   (abort_bkgnd_process_proc.value<>0) and
1124:D  2   (HPIBcheck_sc_proc.value<>0) and
1125:D  2   (HPIBget_amigo_ident_proc.value<>0);
1126:C  2 end; (init_scanstuff)
1127:S
1128:S
1129:C  1 end; (scanstuff)

```



```

1130:D 1 $page$
1131:S
1132:D 1 {program ctable}
1133:S
1134:D 1
1135:D 1 {*****}
1136:D 1 {* Caution: *}
1137:D 1 {* Modify this section only if the desired *}
1138:D 1 {* configuration cannot be achieved by *}
1139:D 1 {* modifying the OPTIONS module. *}
1140:D 1 {*****}
1141:S
1142:D 1 import
1143:S
1144:D 1 const
1145:D 1 srmsysprefix = '/WORKSTATIONS/SYSTEM';
1146:D 1 null_dav =
1147:D 1 dav_type[sc: -1, ba: -1, du: -1, dv: -1];
1148:D 1 null_MSUS =
1149:D 1 MSUS_type[flpy_flags: assign_neither_flpy_unit, letter: NODEVICE, dav: null_dav];
1150:D 1 HP9885_default_MSUS =
1151:D 1 MSUS_type[flpy_flags: assign_neither_flpy_unit, letter: HP9885, dav: HP9885_default_dav];
1152:D 1 MSUS_array_size = 10;
1153:S
1154:D 1 type
1155:D 1 MSUS_array_type = array [1..MSUS_array_size] of MSUS_type;
1156:D 1 log_MSUS_options = (search_for_oTher_uNits, do_not_search_for_oTher_uNits);
1157:S
1158:D 1 var
1159:D -60 1 flpy_MSUS: MSUS_array_type;
1160:D -120 1 harddisc_MSUS: MSUS_array_type;
1161:D -180 1 CS80tape_MSUS: MSUS_array_type;
1162:D -186 1 scanner_MSUS: MSUS_type;
1163:S
1164:D -190 1 local_printer_dav: dav_type;
1165:D -194 1 SRM_dav: dav_type;
1166:D -198 1 BUBBLE_dav: dav_type;
1167:S
1168:D -206 1 index, bus_address, i, nvols: shortint;
1169:D -212 1 lun1, lun2: unitnum;
1170:D -214 1 CS80dt: byte;
1171:D -218 1 CS80id: integer;
1172:D -220 1 CS80hardvols: shortint;
1173:D -228 1 mp: mp_type;
1174:D -234 1 pp: pp_type;
1175:D -235 1 ok: boolean;
1176:S
1177:S
1178:D 1 function increment_and_test_lun: boolean;
1179:C 2 begin {increment_and_test_lun}
1180:C 2 lun := lun+1;
1181:C 2 increment_and_test_lun := lun<=last_harddisc_lun;
1182:C 2 end; {increment_and_test_lun}

```

```

1183:D -235 1 $page$
1184:S
1185:D -122 1 function unit_prefix_successful(dirname: fid): boolean;
1186:D -122 2 var
1187:D -126 2 unitnum: integer;
1188:D -144 2 kvid: vid;
1189:C 2 begin {unit_prefix_successful}
1190:C 2 doprefix(dirname, kvid, unitnum, true);
1191:C 2 unit_prefix_successful := ioreult=ord(ioerror);
1192:C 2 end; {unit_prefix_successful}
1193:S
1194:S
1195:D 1 procedure zero_out_NA_fields(var device_MSUS: MSUS_type);
1196:D 2 const
1197:D 2 clear = true;
1198:D 2 retain = false;
1199:D 2 procedure zero_fields(sc, ba, du, dv: boolean);
1200:C 3 begin {zero_fields}
1201:C 3 if sc then device_MSUS.dav.sc := 0;
1202:C 3 if ba then device_MSUS.dav.ba := 0;
1203:C 3 if du then device_MSUS.dav.du := 0;
1204:C 3 if dv then device_MSUS.dav.dv := 0;
1205:C 3 end; {zero_fields}
1206:C 2 begin {zero_out_NA_fields}
1207:C 2 case device_MSUS.letter of
1208:C 3 INTERNAL:
1209:C 3 zero_fields({sc} clear, {ba} clear, {du} retain, {dv} clear);
1210:C 3 HP9885:
1211:C 3 zero_fields({sc} retain, {ba} clear, {du} retain, {dv} clear);
1212:C 3 HP9885, HP8290X, HP913X_A, HP913X_B, HP913X_C, SRM:
1213:C 3 zero_fields({sc} retain, {ba} retain, {du} retain, {dv} clear);
1214:C 3 BUBBLE:
1215:C 3 zero_fields({sc} retain, {ba} clear, {du} clear, {dv} clear);
1216:C 3 EPROM:
1217:C 3 zero_fields({sc} clear, {ba} clear, {du} clear, {dv} retain);
1218:C 3 otherwise {includes CS80}
1219:C 3 {do nothing};
1220:C 3 end; {case}
1221:C 2 end; {zero_out_NA_fields}
1222:S
1223:S
1224:D 1 procedure assign_flpy_unit_pair(lun: unitnum; dam: ds_type; index: shortint);
1225:C 2 begin {assign_flpy_unit_pair}
1226:C 2 with flpy_MSUS[index], flpy_flags, dav do
1227:C 3 begin
1228:C 3 if assign_even_unit then
1229:C 4 begin
1230:C 4 tea_flpy(lun, letter, dam, sc, ba, du);
1231:C 4 lun := lun+1;
1232:C 4 end; {if}
1233:C 3 if assign_odd_unit then
1234:C 4 tea_flpy(lun, letter, dam, sc, ba, du+1);
1235:C 3 end; {with}
1236:C 2 end; {assign_flpy_unit_pair}

```



```

1237:D -235 1 $page$
1238:S
1239:D -6 1 procedure log_MSUS(MSUS: MSUS_type; log_MSUS_option: log_MSUS_options);
1240:S
1241:S
1242:D -6 2 type
1243:D -6 2 log_flypy_MSUS_options = (assign_both_units, assign_only_this_unit);
1244:S
1245:S
1246:D 2 procedure log_specific_MSUS(var specific_MSUS: MSUS_array_type);
1247:D 3 var
1248:D 3 index: shortint;
1249:D 3 found: boolean;
1250:C -3 3 begin (log_specific_MSUS)
1251:C 3 index := 0;
1252:C 3 repeat
1253:C 4 index := index+1;
1254:C 4 found := MSUSs_match(specific_MSUS[index], MSUS);
1255:C 4 until found or (index=MSUS_array_size);
1256:C 3 if found
1257:C 4 then MSUS.flypy_flags := specific_MSUS[index].flypy_flags (preserve)
1258:C 4 else MSUS.flypy_flags := assign_neither_flypy_unit; (initialize)
1259:C 3 while index>1 do
1260:C 4 begin
1261:C 4 specific_MSUS[index] := specific_MSUS[index-1];
1262:C 4 index := index-1;
1263:C 4 end; (while)
1264:C 3 specific_MSUS[1] := MSUS;
1265:C 3 end; (log_specific_MSUS)
1266:S
1267:S
1268:D 2 procedure log_flypy_MSUS(log_flypy_MSUS_option: log_flypy_MSUS_options);
1269:D 3 var
1270:D 3 odd_unit: boolean;
1271:C -1 3 begin (log_flypy_MSUS)
1272:C 3 with MSUS.dav do
1273:C 4 begin (since floppy units are assigned in pairs...)
1274:C 4 odd_unit := odd(du); (remember which unit this actually is...)
1275:C 4 du := du-odd(odd_unit); (but log only the even-numbered unit!)
1276:C 4 end; (with)
1277:C 3 log_specific_MSUS(flypy_MSUS);
1278:C 3 with flypy_MSUS[1] do (update the flypy_flags)
1279:C 4 if log_flypy_MSUS_option=assign_both_units then
1280:C 5 flypy_flags := assign_both_flypy_units
1281:C 5 else (set only this unit's assignment flag)
1282:C 6 if odd_unit
1283:C 6 then flypy_flags.assign_odd_unit := true
1284:C 6 else flypy_flags.assign_even_unit := true;
1285:C 3 end; (log_flypy_MSUS)
1286:S
1287:S
1288:D 2 procedure log_harddisc_MSUS;
1289:C 3 begin (log_harddisc_MSUS)
1290:C 3 MSUS.dav.dv := 0; (all vols will be assigned, so log only volume zero)
1291:C 3 log_specific_MSUS(harddisc_MSUS);
1292:C 3 end; (log_harddisc_MSUS)

```

```

1293:D -6 2 $page$
1294:S
1295:D 2 function any_9895_unit_missing: boolean;
1296:D 3 var
1297:D -4 3 temp_dav: dav_type;
1298:D -5 3 unit_missing: boolean;
1299:C 3 begin (any_9895_unit_missing)
1300:C 3 temp_dav := MSUS.dav;
1301:C 3 temp_dav.dv := 3; (start with unit 3 and work down)
1302:C 3 repeat (see if all four units are present)
1303:C 4 unit_missing := scanneddevice_letter(temp_dav)<>HP9895;
1304:C 4 temp_dav.dv := temp_dav.dv-1;
1305:C 4 until (temp_dav.dv<0) or unit_missing;
1306:C 3 any_9895_unit_missing := unit_missing;
1307:C 3 end; (any_9895_unit_missing)
1308:S
1309:S
1310:D 2 procedure search_higher_numbered_CS80_units;
1311:D 3 var
1312:D -6 3 temp_MSUS: MSUS_type;
1313:C 3 begin (search_higher_numbered_CS80_units)
1314:C 3 temp_MSUS := MSUS;
1315:C 3 with temp_MSUS.dav do
1316:C 4 if dv<14 then (potentially there are higher-numbered units)
1317:C 5 begin
1318:C 6 dv := dv+1;
1319:C 6 dv := 0; (always look for volume 0)
1320:C 6 letter := scanneddevice_letter(dav);
1321:C 6 log_MSUS(temp_MSUS, search_for_other_units); (recurse!)
1322:C 5 end; (if)
1323:C 3 end; (search_higher_numbered_CS80_units)
1324:S
1325:S
1326:D 2 procedure log_CS80_MSUS;
1327:D 3 var
1328:D -2 3 CS80dt: byte; (device type)
1329:D -3 3 CS80id: integer; (HP product number)
1330:D -3 3 CS80hardvols: shortint; (number of volumes)
1331:D -13 3 CS80mp: mp_type; (media parameters)
1332:D -13 3 const
1333:D -13 3 tape_dt = 2;
1334:D -13 3 min_hd_size = 10000000; (bytes)
1335:C 3 begin (log_CS80_MSUS)
1336:C 3 get_CS80_parms(MSUS.dav, CS80dt, CS80id, CS80hardvols, CS80mp);
1337:C 3 if CS80dt=tape_dt then
1338:C 4 log_specific_MSUS(CS80tape_MSUS)
1339:C 4 else if (CS80hardvols=1) and (CS80mp.bpt*CS80mp.tpm<min_hd_size) then
1340:C 5 log_flypy_MSUS(assign_only_this_unit)
1341:C 5 else
1342:C 6 log_harddisc_MSUS;
1343:C 3 end; (log_CS80_MSUS)

```



```

1344:D -6 2 $page$
1345:S
1346:C 2 begin {log_MSUS}
1347:S
1348:C 2 zero_out_NA_fields(MSUS);
1349:S
1350:C 2 case MSUS.letter of
1351:S
1352:C 3 INTERNAL, HP8290X, HP9885:
1353:C 3 log_flopy_MSUS(assign_both_units);
1354:S
1355:C 3 HP913X_A, HP913X_B, HP913X_C:
1356:C 3 log_harddisc_MSUS;
1357:S
1358:C 3 HP9895:
1359:C 3 if any 9895_unit_missing then {ultimately assign only two units }
1360:C 4 log_flopy_MSUS(assign_both_units)
1361:C 4 else {ultimately assign all four units (probably a 913X)}
1362:C 4 log_harddisc_MSUS;
1363:S
1364:C 3 CS80:
1365:C 3 begin
1366:C 3 if log_MSUS_option=search_for_other_units then
1367:C 4 search_higher_numbered_CS80_units;
1368:C 3 log_CS80_MSUS; {distinguishes tapes, floppies, & hard discs!}
1369:C 3 end;
1370:S
1371:C 3 SRM:
1372:C 3 SRM_dav := MSUS.dav;
1373:S
1374:C 3 PRINTER:
1375:C 3 local_printer_dav := MSUS.dav;
1376:S
1377:C 3 BUBBLE:
1378:C 3 BUBBLE_dav := MSUS.dav;
1379:S
1380:C 3 otherwise
1381:C 3 {do nothing};
1382:S
1383:C 3 end; {case}
1384:S
1385:C 2 end; {log_MSUS}

```

```

1386:D -235 1 $page$
1387:S
1388:C 1 begin {ctable}
1389:S
1390:C 1 { various initializations }
1391:S
1392:C 1 call(cleariohook); {init IO cards in case the BOOTROM drivers touched them}
1393:S
1394:C 1 init_scanstuff;
1395:S
1396:C 1 for index := 1 to MSUS_array_size do
1397:C 2 begin
1398:C 2   flopy_MSUS[index] := null_MSUS;
1399:C 2   harddisc_MSUS[index] := null_MSUS;
1400:C 2   CS80tape_MSUS[index] := null_MSUS;
1401:C 2 end; {for}
1402:S
1403:C 1 SRM_dav := SRM_default_dav; {overridden if bootdevice}
1404:S
1405:C 1 BUBBLE_dav := BUBBLE_default_dav; {overridden if bootdevice}
1406:S
1407:C 1 if local_printer_option=HP18
1408:C 2 then local_printer_dav := local_HP18_printer_default_dav {scan may override}
1409:C 2 else local_printer_dav := local_RS232_printer_default_dav; {scan may override}
1410:S
1411:S
1412:C 1 { log the default 9885 floppy pair, since HP-18 scanning won't include it }
1413:S
1414:C 1 log_MSUS(HP9885_default_MSUS, search_for_other_units);
1415:S
1416:S
1417:C 1 { scan the HP-IB's for mass storage devices and possibly a local printer }
1418:S
1419:C 1 with scanner_MSUS, dav do
1420:C 2 for index := 1 to sc_list_length do
1421:C 3 for bus_address := 0 to 7 do
1422:C 4 begin
1423:C 4   sc := sc_list[index];
1424:C 4   ba := bus_address;
1425:C 4   du := 0;
1426:C 4   dv := 0;
1427:C 4   letter := scanneddevice_letter(dav);
1428:C 4   log_MSUS(scanner_MSUS, search_for_other_units);
1429:C 4 end; {for}
1430:S
1431:C 1 { log internal mini if present, since HP-IB scanning didn't include it }
1432:S
1433:C 1 if internal_mini_present then
1434:C 2 log_MSUS(INTERNAL_MSUS, search_for_other_units);
1435:S
1436:S
1437:C 1 { get the bootdevice MSUS & log it }
1438:S
1439:C 1 get_bootdevice_MSUS(bootdev_MSUS);
1440:C 1 zero_out_NA_fields(bootdev_MSUS); {for tea routine comparisons}
1441:C 1 log_MSUS(bootdev_MSUS, do_not_search_for_other_units);
1442:C 1 bootdev_iun := 0; {set otherwise if/when bootdevice is assigned in tea}
1443:C

```



```

1444:C      1 $page$
1445:S
1446:C      1 { Create a temporary table & fill it with dummy entries }
1447:S
1448:C      1 create_temp_untatable;
1449:S
1450:S
1451:C      1 { standard assignemnts: avoid changing }
1452:S
1453:C      1 tea_memory_volume_dam(primary_dam);
1454:S
1455:C      1 tea_crt( 1);
1456:C      1 tea_kbd( 2);
1457:S
1458:C      1 assign_flypy_unit_pair( 3, primary_dam, {flypy_MSUS[] 1});
1459:S
1460:C      1 with SRM_dav do
1461:C      2   tea_srm( 5, sc, ba, du);
1462:S
1463:C      1 with local_printer_dav do
1464:C      2   tea_local_printer( 6, sc, ba, {uvid} 'PRINTER', local_printer_timeout);
1465:S
1466:S
1467:C      1 { optional floppy unit pairs }
1468:S
1469:C      1 for index := 2 to floppy_unit_pairs do
1470:C      2   assign_flypy_unit_pair(7*(index-2)*2, primary_dam, {flypy_MSUS[] index});
1471:S

```

```

1472:C      1 $page$
1473:S
1474:C      1 { local hard discs }
1475:S
1476:C      1 $if true$
1477:S
1478:C      1   lun := first_harddisc_lun-1;
1479:S
1480:C      1   for index := 1 to MSUS_array_size do
1481:C      2     with harddisc_MSUS[index], dav do
1482:C      3       case letter of
1483:S
1484:C      4         HP9895: {9895 ident with all four units present; probably a HP913X}
1485:C      4           for i := 0 to 3 do
1486:C      5             if increment_and_test_lun then
1487:C      6               tea_HP9895(lun, primary_dam, sc, ba, {du} i, {block_offset} 0);
1488:S
1489:C      4         HP913X_A, HP913X_B, HP913X_C:
1490:C      4           begin
1491:C      4             mp := medium_parameters(letter);
1492:C      4             pp := partitioning_parameters(letter);
1493:C      4             nvols := number_vols(mp, pp);
1494:C      4             for i := 0 to nvols-1 do
1495:C      5               if increment_and_test_lun then
1496:C      6                 tea_amigo_sv(lun, primary_dam, sc, ba, du,
1497:C      6                   vol_offset(i, nvols, mp),
1498:C      6                   letter,
1499:C      6                   vol_bytes(i, nvols, mp));
1500:C      4             end;
1501:S
1502:C      4         CS80:
1503:C      4           begin
1504:C      4             pp := partitioning_parameters(letter);
1505:C      4             repeat
1506:C      5               get_CS80_parms(dav, CS80dt, CS80id, CS80hardvols, mp);
1507:C      5               if mp.tpm=1 then {track partitioning info unavailable...}
1508:C      6                 mp := block_boundaries(mp); {will have to fake it!}
1509:C      5               nvols := number_vols(mp, pp);
1510:C      5               for i := 0 to nvols-1 do
1511:C      6                 if increment_and_test_lun then
1512:C      7                   tea_CS80_mv(lun, primary_dam, sc, ba, du, dv,
1513:C      7                     vol_offset(i, nvols, mp),
1514:C      7                     {devid} CS80id,
1515:C      7                     vol_bytes(i, nvols, mp));
1516:C      5                 dv := dv+1;
1517:C      5               until dv>=CS80hardvols;
1518:C      4             end;
1519:S
1520:C      4           otherwise
1521:C      4             {no local hard disc logged};
1522:S
1523:C      4         end; {case}
1524:S
1525:C      1 $end$ { local hard discs }
1526:S
1527:S
1528:C      1 { CS80 tapes }
1529:S
1530:C      1 $if true$
1531:S

```



```

1542:C      1      with CS80tape_MSUS[1], dav do
1543:C      2          if letter=CS80 then
1544:C      3              tea_CS80_sv(41, LIF_dam, sc, ba, du, dv);
1545:S
1546:C      1      with CS80tape_MSUS[2], dav do
1547:C      2          if letter=CS80 then
1548:C      3              tea_CS80_sv(42, LIF_dam, sc, ba, du, dv);
1549:S
1550:C      1      $end$ { CS80 tapes }
1551:S
1552:C      1      { secondary directory access method entries for highest priority floppies }
1553:S
1554:C      1      assign_flpy_unit_pair(43, secondary_dam, {flpy_MSUS[]} 1);
1555:S
1556:C      1      { duplicate entries for prefixing down the SRM }
1557:S
1558:C      1      (*****
1559:C      1      (* NOTE: Additional duplicate SRM entries may be assigned here, then *)
1560:C      1      (* prefixed down below after assigning the temp_unitable. However *)
1561:C      1      (* for correct behavior in assigning the system unit, specifically *)
1562:C      1      (* if booting off the SRM, unit #49 must be the assigned AFTER all *)
1563:C      1      (* the other SRM units have been assigned: *)
1564:C      1      (*****
1565:S
1566:C      1      with SRM_dav do
1567:C      2          begin
1568:C      3              { tea_srm(46, sc, ba, du); {free}
1569:C      3              tea_srm(45, sc, ba, du); {for possible use as the system unit}
1570:C      2          end; {with}
1571:S
1572:C      1      { secondary directory access method entries for additional floppies }
1573:S
1574:C      1      assign_flpy_unit_pair(47, secondary_dam, {flpy_MSUS[]} 2);
1575:S
1576:C      1      assign_flpy_unit_pair(49, secondary_dam, {flpy_MSUS[]} 3);
1577:S

```

```

1578:C      1      $pages$
1579:S
1580:C      1      { templates for "manually" specifying mass storage table entry assignments }
1581:S
1582:C      1      $if false$ { internal minifloppy in a 9828/9836 }
1583:C      2          tea_mini( 3, primary_dam, {du} 0);
1584:C      2          tea_mini( 4, primary_dam, {du} 1);
1585:S
1586:C      1      $end$
1587:S
1588:C      1      $if false$ { HP8290X, HP9121, or the floppy in an HP913X }
1589:C      2          tea_HP8290X( 3, primary_dam, {sc} 7, {ba} 0, {du} 0);
1590:C      2          tea_HP8290X( 4, primary_dam, {sc} 7, {ba} 0, {du} 1);
1591:S
1592:C      1      $end$
1593:S
1594:C      1      $if false$ { HP9895 }
1595:C      2          tea_HP9895( 7, primary_dam, {sc} 7, {ba} 0, {du} 0, {block_offset} 0);
1596:C      2          tea_HP9895( 8, primary_dam, {sc} 7, {ba} 0, {du} 1, {block_offset} 0);
1597:S
1598:C      1      $end$
1599:S
1600:C      1      $if false$ { HP913X (four volume 9895 look-a-like version) }
1601:C      2          for i := 0 to 3 do
1602:C      3              tea_HP9895(11+i, primary_dam, {sc} 7, {ba} 0, {du} i, {block_offset} 0);
1603:S
1604:C      1      $end$
1605:S
1606:C      1      $if false$ { HP913X_A (5 Mbyte single volume version) }
1607:C      2          mp := medium_parameters(HP913X_A);
1608:C      2          nvols := 4;
1609:C      2          for i := 0 to nvols-1 do
1610:C      3              tea_amigo_sv(11+i, primary_dam, {sc} 7, {ba} 0, {du} 0,
1611:C      3                  vol_offset(i, nvols, mp),
1612:C      3                  HP913X_A,
1613:C      3                  vol_bytes(i, nvols, mp));
1614:S
1615:C      1      $end$
1616:S
1617:C      1      $if false$ { HP913X_B (10 Mbyte single volume version) }
1618:C      2          mp := medium_parameters(HP913X_B);
1619:C      2          nvols := 9;
1620:C      2          for i := 0 to nvols-1 do
1621:C      3              tea_amigo_sv(11+i, primary_dam, {sc} 7, {ba} 0, {du} 0,
1622:C      3                  vol_offset(i, nvols, mp),
1623:C      3                  HP913X_B,
1624:C      3                  vol_bytes(i, nvols, mp));
1625:S
1626:C      1      $end$
1627:S
1628:C      1      $if false$ { HP913X_C (15 Mbyte single volume version) }
1629:C      2          mp := medium_parameters(HP913X_C);
1630:C      2          nvols := 14;
1631:C      2          for i := 0 to nvols-1 do
1632:C      3              tea_amigo_sv(11+i, primary_dam, {sc} 7, {ba} 0, {du} 0,
1633:C      3                  vol_offset(i, nvols, mp),
1634:C      3                  HP913X_C,
1635:C      3                  vol_bytes(i, nvols, mp));
1636:S
1637:C      1      $end$
1638:S

```



```

1624:C      1 $page$
1625:S
1626:S      $if false$ { current CS/80 discs "soft" partitioned by the host }
1627:S      { CS80id := 7908; nvol := 16; mp.tpm := 5* 370; mp.bpt := 35*256; {7908}
1628:S      { CS80id := 7911; nvol := 27; mp.tpm := 3* 572; mp.bpt := 64*256; {7911}
1629:S      { CS80id := 7912; nvol := 30; mp.tpm := 7* 572; mp.bpt := 64*256; {7912}
1630:S      { CS80id := 7914; nvol := 30; mp.tpm := 7*1152; mp.bpt := 64*256; {7914}
1631:S      { CS80id := 7933; nvol := 30; mp.tpm := 13*1321; mp.bpt := 92*256; {7933}
1632:S      { CS80id := 7935; nvol := 30; mp.tpm := 13*1321; mp.bpt := 92*256; {7935}
1633:S      { mp := block_boundaries(mp); {override track boundary partitioning}
1634:S      for i := 0 to nvol-1 do
1635:S      tea_CS80_mv(11+i, primary_dam, {sc} 7, {ba} 0, {du} 0, {dv} 0,
1636:S      vol_offset(1, nvol, mp),
1637:S      {devid} CS80id,
1638:S      vol_bytes(1, nvol, mp));
1639:C
1640:S      1 $end$
1641:S
1642:S      $if false$ { current CS/80 discs "hard" partitioned by the device }
1643:S      { CS80hardnvol := 3;
1644:S      for i := 0 to CS80hardnvol-1 do
1645:S      tea_CS80_sv(11+i, primary_dam, {sc} 7, {ba} 0, {du} 0, {dv} i);
1646:C
1647:S      1 $end$
1648:S
1649:S      $if false$ { Command Set/80 floppy }
1650:S      tea_CS80_sv( 3, primary_dam, {sc} 7, {ba} 0, {du} 0, {dv} 0);
1651:C
1652:S      1 $end$
1653:S
1654:S      $if false$ { Command Set/80 tape }
1655:S      tea_CS80_sv(41, LIF_dam, {sc} 7, {ba} 0, {du} 1, {dv} 0);
1656:C
1657:S      1 $end$
1658:S
1659:S      $if false$ { BUBBLE memory }
1660:S      {watch for conflicting uses of unit 42}
1661:S      {8088LE_DAV.SC default is 30 but may have been changed to boot SC}
1662:S      tea_BUBBLE(42,primary_dam,8088LE_dav.SC);
1663:C
1664:S      1 $end$
1665:S
1666:S      $if false$ { EPROM DISC }
1667:S      {watch for conflicting uses of unit 42}
1668:S      tea_EPROM(42,primary_dam,{ sequence number } 0);
1669:C
1670:S      1 $end$
1671:S
1672:C      1 { end of templates }

```

```

1673:C      1 $page$
1674:S
1675:C      1 { assign the new unitable and unitclear all units }
1676:S
1677:C      1 assign_temp_unitable;
1678:S
1679:S
1680:C      1 { prefix the primary and secondary SRM unit entries }
1681:S
1682:C      1 if not unit_prefix_successful('#5:/') then
1683:C      2 {do nothing}; {tries to set up uvid for possible default unit assignment below}
1684:S
1685:C      1 { if not unit_prefix_successful('#46:/?') then zap_assigned_unit(46); {free}
1686:S
1687:C      1 if not unit_prefix_successful('#45:/' + srmsysprefix + srmnode(unitable^'[45].sc)) then
1688:C      2 if not unit_prefix_successful('#45:/' + srmsysprefix) then
1689:C      3 zap_assigned_unit(45);
1690:S
1691:S
1692:C      1 { remove extraneous local hard disc entries if necessary }
1693:S
1694:C      1 lun2 := first_harddisc_lun;
1695:C      1 while lun2 < last_harddisc_lun do
1696:C      2 begin
1697:C      3   lun1 := lun2;
1698:C      3   repeat
1699:C      4     lun2 := lun2+1;
1700:C      4     until (lun2 > last_harddisc_lun) or not on_same_medium(lun1, lun2);
1701:C      4     pp := partitioning_parameters(unitable^[lun1].letter);
1702:C      4     if pp.mnv < 1 then
1703:C      5       remove_extraneous_volumes(lun1, lun2-1);
1704:C      4     end; {while}
1705:S
1706:S
1707:C      1 { assign the system unit }
1708:S
1709:C      1 if specified_system_unit <> 0 then
1710:C      2 ok := sysunit_ok(specified_system_unit)
1711:C      2 else if (bootdev_lun <> 0) and (unitable^[bootdev_lun].umaxbytes > 300000) then
1712:C      3 ok := sysunit_ok(bootdev_lun)
1713:C      2 else {search for a more suitable system unit}
1714:C      3 begin
1715:C      4   index := 0;
1716:C      4   repeat
1717:C      5     index := index+1;
1718:C      5     ok := sysunit_ok(sysunit_list[index]);
1719:C      5     until ok or (index >= sysunit_list_length);
1720:C      5     if not ok then {revert back to boot device, hoping it was identified}
1721:C      6       ok := sysunit_ok(bootdev_lun);
1722:C      5     end; {else}
1723:S
1724:S
1725:C      1 { special case for default unit assignment }
1726:S
1727:C      1 if sysunit=45 then {set the default unit to the primary SRM unit entry}
1728:C      2 dkvid := unitable^[5].uvid;

```



```
1720:C      1  Epage$
1730:S
1731:C      1  { re-open the standard system files }
1732:S
1733:C      1  openfiles;
1734:S
1735:C      1  end. {ctable}
```

No errors. No warnings.
***** Nonstandard language features enabled *****

DC_DRV

Description

DC_DRV provides low-level driver support.

Usage

DC_DRV is used for the following interfaces:

- 98628 (Data Comm)
- 98629 (SRM)

Requirements

DC and the I/O library kernel (IODECLARATIONS, etc.)

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```
1 S
2 S
3 S
4 S (c) Copyright Hewlett-Packard Company, 1983.
5 S All rights are reserved. Copying or other
6 S reproduction of this program except for archival
7 S purposes is prohibited without the prior
8 S written consent of Hewlett-Packard Company.
9 S
10 S
11 S
12 S
13 S
14 S
15 S
16 S
17 S
18 S
19 S
20 S
21 S
22 S
23 S
24 S
25 S
26 S
27 S
```

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government
is subject to restrictions as set forth in
paragraph (b) (3) (8) of the Rights in Technical
Data and Computer Software clause in
DAR 7-104.9(a).

HEWLETT-PACKARD COMPANY
Fort Collins, Colorado *)

0 \$MODCAL ON\$
0 \$PARTIAL EVAL ON\$
0 \$STACKCHECK ON\$
0 \$RANGE OFF\$
0 \$DEBUG OFF\$
0 \$JVFHECK OFF\$

```
28:D 0 $PAGE$
29:D 0 $SEARCH 'IOLIB:KERNEL'$
30:S
31:D 0
32:D 0
33:D 0
34:D 0
35:D 0
36:D 0
37:D 0
38:D 0
39:D 0
40:D 0
41:D 0
42:D 0
43:D 0
44:D 0
45:D 0
46:D 0
47:D 0
48:D 0
49:D 0
50:D 0
51:D 0
52:D 0
53:D 0
54:D 0
55:D 0
56:D 0
57:D 0
58:D 0
59:D 0
60:D 0
```

RELEASED	VERSION
3.0	

IOLIB	DC_DRIVERS
library	- IOLIB
name	- DC_DRIVERS
module(s)	- init_dc
	- extd
author	-
phone	-
date	- Oct 20, 1981
update	- Aug 11, 1983
release	- ?????????????
source	- IOLIB:DC_DRV.TEXT
object	- IOLIB:DC_DRV.CODE


```

61:D 0 $PAGE$
62:D 0 (*****
63:D 0 *
64:D 0 *
65:D 0 *      BUG FIX HISTORY      - after release 1.0
66:D 0 *
67:D 0 *
68:D 0 *      BUG #    BY / ON      LOC      DESCRIPTION
69:D 0 *      ----    - - - - -    - - - - -
70:D 0 *
71:D 0 *      1283     01/08/82     idc_wtc   IOCONTROL(s,0,1) gives
72:D 0 *                                     error, but should work.
73:D 0 *                                     IOCONTROL(s,513,x)
74:D 0 *                                     would give an error.
75:D 0 *
76:D 0 *      zzzz     06/16/82     idc_init   An error during init
77:D 0 *                                     can blow away the work
78:D 0 *                                     station.
79:D 0 *
80:D 0 *      bbbb     07/12/82     init_dc    No bug sheet.
81:D 0 *                                     Setting up a machine
82:D 0 *                                     number ID if card is a
83:D 0 *                                     Ganglia.
84:D 0 *
85:D 0 *      cccc     08/16/82     init_dc    No bug sheet.
86:D 0 *                                     Hunting for 98629 and
87:D 0 *                                     98628 dsnd1.
88:D 0 *                                     See IODECLARATIONS also.
89:D 0 *
90:D 0 *      367      09/22/82     dc_initialize Allow eXecute of driver
91:D 0 *                                     and have it install
92:D 0 *                                     itself in the system.
93:D 0 *
94:D 0 *      uuuu     09/28/82     init_dc    Differentiate between
95:D 0 *                                     628 and 629 card for
96:D 0 *                                     the card type.
97:D 0 *                                     See IODECLARATIONS also.
98:D 0 *
99:D 0 *      -----    init_dc    Remove machine # setup
100:D 0 (*****

```

```

101:D 0 $PAGE$
102:D 0 (*****
103:D 0 *
104:D 0 *
105:D 0 *      This is the source code for an external procedures library
106:D 0 *      to be used for general purpose interfacing on the HP 9826.
107:D 0 *
108:D 0 *      The library consists of 3 primary sets of modules -
109:D 0 *
110:D 0 *          1.   KERNEL modules
111:D 0 *          2.   driver modules
112:D 0 *          3.   IOLIB modules
113:D 0 *
114:D 0 *      The KERNEL modules consist of the following modules -
115:D 0 *
116:D 0 *          1.   iodeclarations ( contains static r/w space )
117:D 0 *          2.   iocomasm
118:D 0 *          3.   general_0      ( initialization & low level
119:D 0 *                               routines like ioread/iowrite)
120:D 0 *
121:D 0 *      The KERNEL modules also have an executable program segment
122:D 0 *      that gets executed at the time it is loaded. This program
123:D 0 *      initializes the static read/write memory. This program also
124:D 0 *      allocates the temporary storage for any card that exists -
125:D 0 *      independent of whether there is or is not a driver for it.
126:D 0 *
127:D 0 *      The driver modules consist of the actual assembly or PASCAL
128:D 0 *      routines that deal with a specific interface card. There is
129:D 0 *      also an executable program segment for each driver module.
130:D 0 *      This program searches the select code table in the static r/w
131:D 0 *      initialized by the KERNEL general_0 module for all select codes
132:D 0 *      that have the right interface card ( HP18 drivers will search
133:D 0 *      for the 98624 interface ). This program will then set up the
134:D 0 *      driver tables to point to the correct drivers.
135:D 0 *
136:D 0 *      The rest of the IOLIB modules are high-level modules that are
137:D 0 *      used by an end user in his/her application program.
138:D 0 *
139:D 0 *      The KERNEL and some set of driver modules will exist in the
140:D 0 *      SYSTEM.INITLIB file as object code ( not EXPORT text ). The
141:D 0 *      export text will reside on the SYSTEM.LIBRARY file. The rest
142:D 0 *      of the library will reside on the SYSTEM.LIBRARY.
143:D 0 (*****

```



```
144:D      0 $PAGES
145:D      0 (***** )
146:D      0 (*
147:D      0 (*
148:D      0 (*
149:D      0 (*
150:D      0 (*
151:D      0 (*
152:D      0 (*
153:D      0 (*
154:D      0 (*
155:D      0 (*
156:D      0 (*
157:D      0 (*
158:D      0 (*
159:D      0 (*
160:D      0 (*
161:D      0 (*
162:D      0 (*
163:D      0 (*
164:D      0 (*
165:D      0 (*
166:D      0 (*
167:D      0 (*
168:D      0 (*
169:D      0 (*
170:D      0 (***** )
```

REFERENCES :

1. 9826 I/O Designers Guide	(-----)
2. 68000 Manual	(Motorola)
3. Pascal alpha site ERS	(-----)
4. Pascal I/O Library ERS	(-----)
5. 9826 HPL EIO & IOD listings	(-----)
6. 9826 HPL Misc. I/D Doc.	(-----)
7. 9826 card documentation	(Mfg. Specs.)
8. Pascal I/O Library IRS	(-----)
9. 98628 Data comm documentation	(-----)

```
171:D      0 $PAGES
172:D      0 PROGRAM dc_initialize ( INPUT , OUTPUT );
```



```

173:D 1 $PAGES
174:D 1 (*****
175:D 1 (*
176:D 1 (*
177:D 1 (*      DATA COMM DRIVERS
178:D 1 (*
179:D 1 (*
180:D 1 (*****
181:D 1 EXTERNAL MODULE extdc;
182:S   ( by -----
183:S   date 10/20/81
184:S   update 11/03/81
185:S
186:S   purpose This module is a declaration of the importation text for
187:S   the external drivers.
188:S
189:S   note The assembly language code that is imported needs to be
190:S   called 'extdc'. The routines need to be called
191:S   'extdc_@@@@'.
192:D 1   )
193:S
194:D 1 IMPORT sysglobals, iodeclarations;
195:S
196:D 1 EXPORT
197:S
198:D 1 PROCEDURE alvinit ( temp : ANYPTR );
199:D 1 PROCEDURE alvinisr ( temp : PISRTB );
200:D 1 PROCEDURE enter_data ( temp : ANYPTR ; x : ANYPTR ;
201:D 2 VAR c : INTEGER );
202:D 2 PROCEDURE output_data ( temp : ANYPTR ; x : ANYPTR ;
203:D 2 cnt : INTEGER );
204:D 1 PROCEDURE output_end ( temp : ANYPTR );
205:D 1 PROCEDURE direct_status ( temp : ANYPTR ; reg : io_word;
206:D 2 VAR x : io_word);
207:D 2 PROCEDURE direct_control ( temp : ANYPTR ; reg : io_word;
208:D 2 val : io_word );
209:D 2 PROCEDURE control_bfd ( temp : ANYPTR ; reg : io_word;
210:D 2 val : io_word );
211:D 1 PROCEDURE start_tfr_in ( temp : ANYPTR );
212:D 1 PROCEDURE start_tfr_out ( temp : ANYPTR );
213:S
214:D 1 END; ( of extdc )

```

```

215:D 1 $PAGES
216:D 1 (*****
217:D 1 (*
218:D 1 (*
219:D 1 (*      DATA COMM DRIVERS
220:D 1 (*
221:D 1 (*
222:D 1 (*****
223:D 1 MODULE intdc;
224:S   ( by -----
225:S   date 10/20/81
226:S   update 08/16/82
227:S
228:S   purpose This module contains the internal drivers.
229:S
230:D 1   )
231:S
232:D 1 IMPORT sysglobals,
233:D 1 iodeclarations ;
234:S
235:D 1 EXPORT
236:S
237:D 1 TYPE dc_err_type = PACKED ARRAY
238:D 1 [minrealisc..maxrealisc] OF io_word ; { zzzz TM 6/16/82 }
239:D 1 dc_err_ptr = ^dc_err_type ; { zzzz TM 6/16/82 }
240:S
241:D -1 1 VAR dc_init_fault : BOOLEAN ; { zzzz TM 6/16/82 }
242:D -6 1 dc_error : dc_err_ptr ; { zzzz TM 6/16/82 }
243:S
244:D -6 1 PROCEDURE idc_init ( temp : ANYPTR );
245:D -6 1 PROCEDURE idc_isr ( temp : PISRTB );
246:D -6 1 PROCEDURE idc_rdb ( temp : ANYPTR ; VAR x : CHAR);
247:D -6 1 PROCEDURE idc_wtb ( temp : ANYPTR ; val : CHAR);
248:D -6 1 PROCEDURE idc_rdw ( temp : ANYPTR ; VAR x : io_word);
249:D -6 1 PROCEDURE idc_wtw ( temp : ANYPTR ; val : io_word);
250:D -6 1 PROCEDURE idc_rds ( temp : ANYPTR ; reg : io_word;
251:D 2 VAR x : io_word);
252:D -6 2 PROCEDURE idc_wtc ( temp : ANYPTR ; reg : io_word;
253:D -6 2 val : io_word);
254:D -6 1 PROCEDURE idc_tfr ( temp : ANYPTR ; bcb : ANYPTR );
255:S
256:D -6 1 IMPLEMENT
257:S
258:S
259:D -6 1 IMPORT isr,
260:D -6 1 general_0,
261:D -6 1 extdc ;
262:S
263:D 1 PROCEDURE idc_init ( temp : ANYPTR );
264:D 2 VAR dummyword : io_word;
265:D -6 2 dummy_isc : INTEGER; { bbbb TM 7/12/82 }
266:C 2 BEGIN
267:C 2 control_bfd ( temp, 0, 1 );
268:S
269:C 2 dummy_isc := io_find_isc(temp); { bbbb TM 7/12/82 }
270:S
271:C 2 ( make sure card is still async )
272:S
273:C 2 direct_status( temp, 3, dummyword );
274:C 2 WITH isc_table[ dummy_isc ] DO BEGIN

```



```

275:      card_id := hp_datacomm;
276:      IF dummyword = 1 THEN card_id := hp98628_async;
277:      IF dummyword = 2 THEN card_id := hp98628_dsnd1;
278:      END; { of WITH DO BEGIN }
279:
280:      IF dc_init_fault
281:      THEN BEGIN
282:          IF dc_error^[dummy_isc]<>0
283:          THEN BEGIN
284:              io_escape(dc_error^[dummy_isc],
285:                  dummy_isc);
286:          END; { of IF dc_error<> 0 }
287:      END; { of IF dc_init_fault }
288:
289:      { set up Ganglia card ID }
290:      IF ioread_byte(dummy_isc,HEX('402F'))=3
291:      THEN BEGIN
292:          { card is ganglia }
293:          { isc table( dummy_isc ).card_id := hp98629;
294:          { iowrite_byte(dummy_isc
295:          {   HEX('4061'), io_model_number DIV 256 );
296:          { iowrite_byte(dummy_isc
297:          {   HEX('4061') + 2, io_model_number MOD 256 );
298:          END; { of IF }
299:
300:
301:      2  END; { of idc_init }
302:
303:
304:
305:      1  PROCEDURE idc_isr ( temp : PISIRB );
306:      BEGIN
307:          alvin_isr( temp );
308:      END; { of idc_isr }
309:
310:
311:
312:      1  PROCEDURE idc_rdb ( temp : ANYPTR ; VAR x : CHAR);
313:      VAR count : INTEGER;
314:      BEGIN
315:          { this can escape with an eod escape }
316:          count := 1;
317:          enter_data( temp , ADDR(x) , count );
318:      END; { of idc_rdb }
319:
320:
321:
322:      1  PROCEDURE idc_wtb ( temp : ANYPTR ; val : CHAR);
323:      BEGIN
324:          { this can escape with an eod escape }
325:          output_data( temp , ADDR(val) , 1 );
326:      END; { of idc_wtb }
327:
328:
329:
330:      1  PROCEDURE idc_rdw ( temp : ANYPTR ; VAR x : io_word);
331:      VAR count : INTEGER;
332:      BEGIN
333:          { this can escape with an eod escape }
334:          count := 2;

```

```

335:      2  enter_data( temp , ADDR(x) , count );
336:      2  END; { of idc_rdw }
337:
338:
339:
340:      1  PROCEDURE idc_wtw ( temp : ANYPTR ; val : io_word);
341:      BEGIN
342:          { this can escape with an eod escape }
343:          output_data( temp , ADDR(val) , 2 );
344:      END; { of idc_wtw }
345:
346:
347:
348:      1  PROCEDURE idc_rds ( temp : ANYPTR ; reg : io_word;
349:      VAR x : io_word);
350:      BEGIN
351:          direct_status( temp , reg , x );
352:      END; { of idc_rds }
353:
354:
355:
356:      1  PROCEDURE idc_wtc ( temp : ANYPTR ; reg : io_word;
357:      VAR dummyword : io_word;
358:      VAR val : io_word);
359:      BEGIN
360:          { range of valid registers -
361:          000-127 buffered control
362:          257-383 direct control
363:          512 abrt tfr in
364:          513 abrt tfr out
365:
366:
367:      2  }
368:
369:      2  IF ( { reg >= 257 } AND { reg <= 383 } ) OR
370:      3  { reg = 512 } OR { reg = 513 }
371:      3  THEN BEGIN
372:          3  direct_control( temp , reg-256 , val );
373:
374:      3  END
375:      3  ELSE BEGIN
376:          3  IF ( { reg >= 000 } AND { reg <= 127 } )
377:          4  THEN BEGIN
378:              4  control_bfd( temp , reg , val );
379:
380:          4  IF ( reg = 0 )
381:          5  THEN BEGIN
382:              5  { make sure card is still async }
383:
384:              5  direct_status( temp , 3 , dummyword );
385:              5  WITH isc_table[ io_find_isc(temp) ] DO
386:              6  IF dummyword = 1 THEN card_id := hp98628_async
387:              7  ELSE card_id := hp_datacomm;
388:              5  END; { of IF reg=0 }

```



```

395:C      4      END
396:C      4      ELSE BEGIN
397:C
398:C      4      io_escape(ioe_misc,io_find_isc(temp));
399:C
400:C      4      END; { of IF reg<=127 }
401:C
402:C      3      END; { of IF >=257 }
403:C
404:C      2      END; { of idc_wtc }
405:C
406:C
407:C
408:D      1      PROCEDURE idc_tfr ( temp : ANYPTR ; bcb : ANYPTR );
409:D      -4      2      VAR b_info : ^buf_info_type;
410:D      -6      2      io_isc : type_isc ;
411:D      -10     2      tmpptr : pio_tmp_ptr;
412:D      -14     2      tmpcnt : INTEGER ;
413:D      -18     2      mycount : INTEGER ;
414:D      -19     2      done : 800LEAN ;
415:C
416:C      2      BEGIN
417:C      2      b_info := ANYPTR( bcb );
418:C      2      tmpptr := ANYPTR( temp );
419:C      2      io_isc := tmpptr^.my_isc ;
420:C      2      WITH b_info^ DO BEGIN
421:C      3      IF ( usr_tfr = serial_DMA ) OR
422:C      3      { usr_tfr = overlap_DMA } OR
423:C      4      { b_w_mode = TRUE } { word }
424:C      4      THEN BEGIN
425:C      4      { error }
426:C      4      IF direction = from_memory
427:C      3      THEN tmpptr^.out_bufptr := NIL;
428:C      3      ELSE tmpptr^.in_bufptr := NIL;
429:C      4      io_escape( ioe_bad_tfr , io_isc );
430:C      4      END;
431:C
432:C      3      { mark buffer busy }
433:C
434:C      3      active_isc := io_isc ;
435:C
436:C
437:C      3      IF usr_tfr < overlap_INTR
438:C      4      THEN BEGIN
439:C      4      { serial transfer - handled here }
440:C      4      IF direction = from_memory
441:C      5      THEN BEGIN
442:C      5      { serial output tfr }
443:C      5      output_data ( temp , buf_empty , term_count );
444:C      5      buf_empty := ANYPTR( INTEGER(buf_empty) + term_count);
445:C      5      term_count := 0;
446:C
447:C      5      IF end_mode THEN output_end( temp );
448:C
449:C      5      END
450:C      5      ELSE BEGIN
451:C
452:C      5      END
453:C      5      ELSE BEGIN
454:C

```

```

455:C      5      { serial input tfr }
456:C
457:C      5      done := false;
458:C      5      mycount := term_count; { tfr size }
459:C      5      REPEAT
460:C      6      IF term_char = -1
461:C      7      THEN BEGIN
462:C      7      { serial input - no term }
463:C      7      TRY
464:C      8      enter_data( temp , buf_fill , mycount );
465:C      8      term_count := term_count - mycount;
466:C      8      buf_fill := ANYPTR( INTEGER(buf_fill) + mycount);
467:C      8      done := TRUE;
468:C      8      RECOVER BEGIN
469:C      8      IF { escapecode = ioescapecode } AND
470:C      8      { io_result = io_eod_seen } AND
471:C      8      { io_isc = io_isc }
472:C      9      THEN BEGIN
473:C      9      term_count := term_count - mycount;
474:C      9      buf_fill := ANYPTR( INTEGER(buf_fill) + mycount);
475:C      9      mycount := term_count;
476:C      9      IF end_mode
477:C      10     THEN BEGIN
478:C      10     done := TRUE;
479:C      10     END; { of IF end_mode }
480:C      10     END
481:C      10     ELSE BEGIN
482:C      9     escape(escapecode);
483:C      9     END; { of IF }
484:C      9     END; { of TRY RECOVER BEGIN }
485:C      7     ELSE BEGIN
486:C      7     { serial input - termination }
487:C      7     TRY
488:C      7     REPEAT
489:C      8     tmpcnt := 1;
490:C      8     enter_data( temp , buf_fill , tmpcnt );
491:C      8     buf_fill := ANYPTR( INTEGER(buf_fill) + 1 );
492:C      8     term_count := term_count - 1;
493:C      8     UNTIL { term_count = 0 } OR
494:C      8     { CHARPTR( INTEGER(buf_fill)-1 ) = CHR(term_char) };
495:C      8     done := TRUE;
496:C      8     RECOVER BEGIN
497:C      8     IF { escapecode = ioescapecode } AND
498:C      8     { io_result = io_eod_seen } AND
499:C      8     { io_isc = io_isc }
500:C      9     THEN BEGIN
501:C      9     { I use tmpcnt because the eod may or may not have
502:C      9     been caught with the character }
503:C      9     term_count := term_count - tmpcnt;
504:C      9     buf_fill := ANYPTR( INTEGER(buf_fill) + tmpcnt );
505:C      9     IF end_mode
506:C      10     THEN BEGIN
507:C      10
508:C      10
509:C      10
510:C      10
511:C      9
512:C      9
513:C      9
514:C      10

```



```

515:C      10      done := TRUE;
516:C      10      END; { of IF end_mode }
517:C      9      END
518:C      9      ELSE BEGIN
519:C      9      escape(escapecode);
520:C      9      END; { of IF }
521:C      18      END; { of TRY RECOVER BEGIN }
522:C      7      END; { of IF term_char }
523:C      6      UNTIL done;
524:C      5      END; { of IF direction }
525:C      5
526:C      4      { mark buffer not busy }
527:C      4
528:C      4      IF direction = from_memory
529:C      4      THEN tmpptr^.out_bufptr := NIL;
530:C      4      ELSE tmpptr^.in_bufptr := NIL;
531:C      4      active_isc := no_isc;
532:C      4      drv_tmp_ptr := NIL;
533:C      4
534:C      4      IF eot_proc.dummy_pr <> NIL
535:C      4      THEN BEGIN
536:C      4      { call user eot procedure }
537:C      4      CALL ( eot_proc.real_proc, eot_parm );
538:C      4      END; { of IF }
539:C      4
540:C      4      END
541:C      4      ELSE BEGIN
542:C      4      { overlap transfer - handled by drivers }
543:C      4
544:C      4      IF direction = to_memory
545:C      4      THEN BEGIN
546:C      5      start_tfr_in ( temp );
547:C      5
548:C      5      END
549:C      5      ELSE BEGIN
550:C      5      start_tfr_out ( temp );
551:C      5
552:C      5      END; { of IF direction }
553:C      4      END; { of IF overlap }
554:C      3      END; { of WITH b_info }
555:C      2      END; { of idc_tfr }
556:C      1      END; { of intdc }

```

```

566:D      1 $PAGE$
567:D
568:D      1 MODULE init_dc;
569:D      {
570:D      date 10/20/81
571:D      update 08/11/83 vers number only
572:D      purpose This module initializes the data comm drivers.
573:D      }
574:D      1
575:D
576:D      1 IMPORT iodeclarations ;
577:D
578:D      1 EXPORT
579:D
580:D      1 VAR
581:D      -120 1 dc_drivers : drv_table_type;
582:D
583:D      -120 1 PROCEDURE io_init_dc;
584:D
585:D      -120 1 IMPLEMENT
586:D
587:D      -120 1 IMPORT sysglobals ,
588:D      -120 1 isr ,
589:D      -120 1 general_D ,
590:D      -120 1 extdc ,
591:D      -120 1 intdc ;
592:D
593:D      1 PROCEDURE io_init_dc;
594:D      2
595:D      2 VAR
596:D      -2 2 io_isc : type_isc;
597:D      -4 2 dummyword : io_word;
598:D      -6 2 io_lvl : io_byte;
599:D
600:D      2 BEGIN
601:D
602:D      2 io_revid := io_revid + ' 53.0'; { SERIAL revision added 2/5/82 TM }
603:D
604:D      2 { set up the driver tables }
605:D
606:D      2 WITH dc_drivers DO BEGIN
607:D      3 dc_drivers := dummy_drivers ;
608:D      3 idc_init := idc_init;
609:D      3 idc_isr := idc_isr;
610:D      3 idc_rdb := idc_rdb;
611:D      3 idc_wtb := idc_wtb;
612:D      3 idc_rdw := idc_rdw;
613:D      3 idc_wtw := idc_wtw;
614:D      3 idc_rds := idc_rds;
615:D      3 idc_wtc := idc_wtc;
616:D      3 idc_tfr := idc_tfr;
617:D      3 END; { of WITH }
618:D
619:D      2
620:D
621:D
622:D
623:D
624:D
625:D

```



```

628:C      2      { set up drivers for the interfaces }
629:C      2      FOR io_isc:=iominisc TO iomaxisc DO
630:C      3      WITH isc_table[io_isc] DO BEGIN
631:C      4          IF { card_id = hp98628_async } OR
632:C      5          { card_id = hp98628_dsnd1 } OR
633:C      6          { card_id = hp98629 }
634:C      7          THEN BEGIN
635:C      8              card_id := hp_datacomm;
636:C      9              card_type := serial_card;
637:C      5          END;
638:C      4          IF card_id = hp_datacomm
639:C      5          THEN BEGIN
640:C      6              io_drv_ptr:=ADDR(dc_drivers);
641:C      7              { if the card exists then link in an ISR for it }
642:C      8              { ??? - what happens if an ISR fires during init }
643:C      9              io_lvl:=(ioread_byte(io_isc,3) DIV 16) MOD 4)+3;
644:C      6              IF io_tmp_ptr^.myisrib.INTEGRADDR <> NIL
645:C      7              THEN BEGIN
646:C      8                  { if isr exists then unlink it }
647:C      9                  ISRLINK(io_lvl
648:C      6                  ADDR(io_tmp_ptr^.myisrib));
649:C      7                  END; { of IF }
650:C      5                  PERMISRLINK(io_drv_ptr^.iod_isr,
651:C      6                  ANYPTR(INTEGER(card_ptr)+3),
652:C      7                  192,
653:C      8                  192,
654:C      9                  io_lvl,
655:C      6                  ADDR(io_tmp_ptr^.myisrib));
656:C      7                  END; { of IF card_type = hp_datacomm }
657:C      4      END; { of FOR io_isc WITH isc_table[io_isc] BEGIN }
658:C      2      { call the actual driver initialization }
659:C      2      dc_init_fault := FALSE;
660:C      2      FOR io_isc:=iominisc TO iomaxisc DO
661:C      3      WITH isc_table[io_isc] DO
662:C      4      IF card_id = hp_datacomm
663:C      5      THEN BEGIN
664:C      6          TRY
665:C      7              alvinit( io_tmp_ptr );
666:C      8              direct_status( io_tmp_ptr , 3 , dummyword );
667:C      9              IF dummyword = 1 THEN card_id := hp98628_async;
668:C      6              IF dummyword = 2 THEN card_id := hp98628_dsnd1;
669:C      7              { set up Ganglia card ID }
670:C      8              IF ioread_byte(io_isc,HEX('402F'))=3
671:C      9              THEN BEGIN
672:C      6                  { card is ganglia }
673:C      7                  { level info }
674:C      8                  { isr }
675:C      9                  { card address }
676:C      6                  { intr. mask }
677:C      7                  { intr. value }
678:C      8                  { level }
679:C      9                  { isr info }
680:C      6                  { level }
681:C      7                  { isr info }
682:C      8                  { level }
683:C      9                  { isr info }
684:C      6                  { level }
685:C      7                  { isr info }
686:C      8                  { level }
687:C      9                  { isr info }
688:C      6                  { level }
689:C      7                  { isr info }
690:C      8                  { level }
691:C      9                  { isr info }
692:C      6                  { level }
693:C      7                  { isr info }
694:C      8                  { level }
695:C      9                  { isr info }
696:C      6                  { level }
697:C      7                  { isr info }
698:C      8                  { level }
699:C      9                  { isr info }
700:C      6                  { level }
701:C      7                  { isr info }
702:C      8                  { level }
703:C      9                  { isr info }
704:C      6                  { level }
705:C      7                  { isr info }
706:C      8                  { level }
707:C      9                  { isr info }
708:C      6                  { level }
709:C      7                  { isr info }
710:C      8                  { level }
711:C      9                  { isr info }
712:C      6                  { level }
713:C      7                  { isr info }
714:C      8                  { level }
715:C      9                  { isr info }

```

```

686:C      7      card_id := hp98629;
687:C      7      card_type := srm_card;
688:C      7      { iowrite_byte(io_isc ,
689:C      8      { HEX('4061') , io_model_number DIV 256 );
690:C      7      { iowrite_byte(io_isc ,
691:C      8      { HEX('4061') + 2 , io_model_number MOD 256 );
692:C      7      END; { of IF }
693:C      6      RECOVER BEGIN
694:C      7      IF ESCAPECODE<>ioescapecode THEN ESCAPE(ESCAPECODE);
695:C      6      IF io_isc <> io_isc THEN ESCAPE(ESCAPECODE);
696:C      7      IF NOT dc_init_fault
697:C      8      THEN BEGIN
698:C      9          dc_init_fault := TRUE;
699:C      6          NEW(dc_error);
700:C      7          FOR dummyword:=minrealisc TO maxrealisc DO
701:C      8          dc_error^[dummyword]:=0;
702:C      9          END; { of IF NOT dc_init_fault }
703:C      6          dc_error^[io_isc] := io_result;
704:C      7          END; { of RECOVER BEGIN }
705:C      5          END; { of WITH IF }
706:C      2      END; { of io_init_dc }
707:C      1      END; { of MODULE init_dc }

```



```
716:D      ; $PAGE$
717:S
718:D      ; IMPORT    init_dc ,
719:D      1          LOADER ;
720:S                                     ( 367  TM 9/22/82 )
721:S
722:C      1 BEGIN
723:C      1   io_init_dc;
724:C      1   MARKUSER;
725:C      1 END. ( of dc_initialize )
                                     ( 367  TM 9/22/82 )
```

No errors. No warnings.

***** Nonstandard language features enabled *****

DGL_C_IN

Description

DGL_C_IN provides selection of different input device handlers.

Requirements

GLE_TYPES, GLE_HPGL_IN, GLE_UTLS, GLE_HPIB_IO, GENERAL_O, IODECLARATIONS, SYSGLOBALS, IOCOMASM, GLE_KNOB_IN, DGL_TOOLS, DGL_VARS, DGL_KNOB, DGL_TYPES, GLE_GEN, GLE_GENI, DGL_GEN, and DGL_CONFIG_OUT.

Notes

See also LIB.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

12067:0      1 procedure termhpgl ( anyvar iocb_ptr : anyptr );
12068:S
12069:0      2 var
12070:0      -4 2 iocb_ptr_hpib : hpib_iocb_ptr;
12071:0      -8 2 buf : ascii_buffer_ptr;
12072:S
12073:0      2 begin
12074:0      2 with gle_gcbl^ do
12075:0      3 begin
12076:0      3 hpib_term(iocb_ptr);           { perform io term then release mem }
12077:0      3 iocb_ptr_hpib := iocb;
12078:0      3 {dispose(iocb_ptr_hpib);}
12079:0      3 buf := device_buf;
12080:0      3 {dispose(buf);}
12081:0      3 end;
12082:0      2 end;
12083:S
12084:0      1 procedure setuphpgl ( gcbl : graphics_input_control_block_ptr );
12085:S
12086:0      2 var
12087:0      -4 2 iocb_ptr_hpib : hpib_iocb_ptr;
12088:0      -8 2 buf : ascii_buffer_ptr;
12089:0      -10 2 cnt : gle_shortint;
12090:0      -14 2 address : integer;
12091:S
12092:0      2 begin
12093:0      2 with gcbl^ do
12094:0      3 begin
12095:0      3 error_return := 1;
12096:0      3 try
12097:0      4 address := gle_read_integer(device_info_char_count,device_info,cnt);
12098:S
12099:0      4 buf := addr(ascii_buffer_space);
12100:0      4 device_buf := buf;
12101:0      4 iocb_ptr_hpib := addr(hpib_iocb_space);
12102:0      4 iocb := iocb_ptr_hpib;
12103:S
12104:0      4 io_write := hpib_write;
12105:0      4 io_read := hpib_read;
12106:0      4 io_term := termhpgl;
12107:0      4 io_inq_timeout := hpib_inq_timeout;
12108:0      4 io_set_timeout := hpib_set_timeout;
12109:0      4 with iocb_ptr_hpib^ do
12110:0      5 begin
12111:0      5 device_addr := device_info;
12112:0      5 name_size := device_info_char_count;
12113:0      5 hpib_init ( iocb_ptr_hpib );
12114:0      5 error_return := 0;
12115:0      5 if error = 0 then gle_init_hpgl_input (gcbl)
12116:0      5 else error_return := 1;
12117:0      5 if error_return = 0 then dgl_hpgli_init
12118:0      5 { otherwise clean up the hpib bus ( 2.1 bug fix ) }
12119:0      5 else hpib_init ( iocb_ptr_hpib );
12120:0      5 end;
12121:0      4 recover
12122:0      4 { ignore io, and value range errors }
12123:0      4 if (escapecode <> -26) and (escapecode <> -8) then escape(escapecode);
12124:S      3 if error_return <> 0 then
12125:0      3 begin
12126:S      {dispose(iocb_ptr_hpib); dispose(buf); { clean up }

```

```

12127:0      3 end;
12128:0      3 end;
12129:0      2 end;
12130:S
12131:0      1 procedure configure_input_gle ( gcbl : graphics_input_control_block_ptr );
12132:S
12133:0      2 begin
12134:0      2 setupknob ( gcbl );
12135:0      2 if gcbl^ error_return <> 0 then setuphpgl ( gcbl );
12136:0      2 end;
12137:S
12138:0      1 end. { of module }
12139:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

DGL_C_OUT

Description

DGL_C_OUT provides selection of different output device handlers.

Requirements

GLE_TYPES, SYSDEVS, SYSGLOBALS, GLE_HPGL_OUT, GLE_STEXT, GLE_ASTEXT, GLE_ASCLIP, GLE_SCLIP, GLE_SMARK, GLE_AUTL, GLE_UTLS, GLE_RAS_OUT, GLE_ARAS_OUT, GLE_FILE_IO, GLE_HPIB_IO, GENERAL_O, IODECLARATIONS, IOCOMASM, DGL_TOOLS, DGL_RASTER, DGL_TYPES, DGL_VARS, ASM, LOADER, GLE_GEN, DGL_GEN, and DGL_HPGL.

Notes

See also LIB.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:D      0 {
2:D      0 { Graphics Library
3:D      0 {
4:D      0 { Module = DGL_CONFIG_OUT
5:D      0 { Programmer = 835
6:D      0 { Date = 10- 5-82
7:D      0 {
8:D      0 { Purpose: To link device dependent drivers with the graphics library.
9:S      0 {
10:D     0 { Rev history
11:D     0 { Created - 10- 5-82
12:D     0 { Modified - 1-12-84 BDS -Added Gator black-white support
13:D     0 { Modified - 2-17-84 BDS -Changed dynamic to global storage for PASC 3.0
14:S     0 {
15:S     0 { (c) Copyright Hewlett-Packard Company, 1983.
16:S     0 { All rights are reserved. Copying or other
17:S     0 { reproduction of this program except for archival
18:S     0 { purposes is prohibited without the prior
19:S     0 { written consent of Hewlett-Packard Company.
20:S     0 {
21:S     0 {
22:S     0 {
23:S     0 {
24:S     0 {
25:S     0 {
26:S     0 {
27:S     0 {
28:S     0 {
29:S     0 {
30:S     0 {
31:D     0 { HEWLETT-PACKARD COMPANY
32:S     0 { Fort Collins, Colorado
33:D     0 {
34:D     0 { $SEARCH 'GLE_LIB',
35:D     0 { 'TYPES',
36:D     0 { 'DGL_VARS',
37:D     0 { 'DGL_TOOLS',
38:D     0 { 'DGL_RAS',
39:D     0 { 'DGL_HPGL',
40:S     0 { $modcals
41:D     0 { $include 'OPTIONS'$ { compiler options }
42:D     0 {
43:D     0 {
44:D     0 {
45:D     0 {
46:D     0 {
47:D     0 {
48:D     0 {
49:S     0 {
50:S     0 {
51:S     0 {
52:D     0 { $include 'OPTIONS'$ { compiler options }
53:D     0 { $LINENUM 11000$
54:S     0 {
55:D     0 { module DGL_CONFIG_OUT;
56:D     1 { import gle_types, sysdevs;
57:D     1 { export

```

```

11007:D 1 procedure configure_gle ( gcb : graphics_control_block_ptr );
11008:S
11009:D 1 implement
11010:S
11011:D 1 import gle_hppl_out, { hppl plotter support }
11012:D 1 gle_ras_out, { raster support }
11013:D 1 gle_file_io, { plotter spooling io }
11014:D 1 gle_hpib_io, { plotter HPiB support }
11015:D 1 gle_utils, { general tools }
11016:D 1 dgl_tools, { used to get machine type }
11017:D 1 sysglobals, { address for GRAPHICSBASE }
11018:D 1 dgl_declarations, { used to get min, max selectcode ranges }
11019:D 1 dgl_raster, { DGL device dependent raster init code }
11020:D 1 dgl_hppl, { DGL device dependent HPGL init code }
11021:D 1 dgl_vars; { DGL global data }
11022:D 1 var
11023:D -8 1 save_crthook: crtlltype;
11024:D -12 1 hp98527A_address : anyptr; { holds adr of first graphics plane }
11025:D -13 1 found_gator: boolean;
11026:D -16 1 select_code : shortint;
11027:D -17 1 has_color : boolean;
11028:D -22 1 frame_buffer : integer;
11029:D -26 1 stat : ^shortint;
11030:D -30 1 ptr : ^shortint;
11031:D -32 1 int_ext_gator : shortint; { 0=neither, 1=internal, 2=external }
11032:D -550 1 raster_device_rec_space : raster_device_rec;
11033:D -852 1 hppl_device_rec_space : hppl_device_rec;
11034:D -816 1 ascii_buffer_space : ascii_buffer;
11035:D -1488 1 file_iocb_space : file_iocb;
11036:D -1506 1 hpib_iocb_space : hpib_iocb;
11037:D -1507 1 took_type_ahead : boolean;
11038:D -1508 1 reduced_screen : boolean;
11039:D -1509 1 secondary : boolean;
11040:D -1510 1 moon : boolean;
11041:S
11042:S {procedure dummy_crthook(dum_op : crtlltype; anyvar dum_pos : integer;
11043:S { dum_c : char);
11044:S begin
11045:D -1510 1 end;
11046:S
11047:D 1 procedure termraster ( anyvar iocb_ptr : anyptr );
11048:S
11049:D 2 var
11050:D -4 2 ptr : raster_device_rec_ptr;
11051:D -5 2 charvar : char;
11052:S
11053:C 2 begin
11054:C 2 with gle_gcb^ do
11055:C 2 begin
11056:C 3 ptr := dev_dep_stuff;
11057:C 3 if reduced_screen then
11058:C 4 begin
11059:C 4 with gcb^, raster_device_rec_ptr(dev_dep_stuff)^ do
11060:C 5 begin
11061:C 5 reduced_screen := true;
11062:C 5 n_lines := 752;
11063:C 5 hard_ymax := 751;
11064:C 5 end;
11065:C 4 end;
11066:C 3 if took_type_ahead then

```



```

11067:C      4      begin
11068:C      4      crtllhook := save_crtllhook;
11069:C      4      keybufops(kdisplay,charvar);
11070:C      4      end;
11071:C      3      (dispose(ptr));
11072:C      3      end;
11073:C      2 end;
11074:S
11075:S
11076:S      {Look for gator present. Assume only one gator is on the bus and that
11077:D -1510 1 an internal one overrides an external one if multiples present.}
11078:D
11079:D      1 procedure gatorcrttype(var found_gator,has_color:boolean;
11080:D      2 var frame_buffer:integer;
11081:D      2 var select_code,int_ext_gator :shortint);
11082:S
11083:D      2 const
11084:D      2 gatorid=25;
11085:D      2 var
11086:D      2 i: shortint;
11087:D      2 dummy: shortint;
11088:S
11089:S
11090:C      2 begin
11091:C      2 control_space:=0;
11092:C      2 found_gator:=false;
11093:C      2 int_ext_gator := 0;
11094:C      2 ptr:=anyptr(hex('560000'));
11095:S
11096:C      2 try
11097:C      3 dummy:=ptr^;
11098:C      3 if (dummy mod 128) = gatorid then (look for internal gator )
11099:C      4 begin
11100:C      4 found_gator:=true;
11101:C      4 control_space:=integer(ptr);
11102:C      4 end;
11103:C      3 recover
11104:C      3 if escapecode<>-12 then escape(escapecode);
11105:S
11106:C      2 if found_gator then (if there, find frame buffer)
11107:C      3 begin
11108:C      3 int_ext_gator := 1;
11109:C      3 stat := anyptr(control_space + 16384);
11110:C      3 frame_buffer := ((stat^ mod 16)*1048576;
11111:C      3 end
11112:S
11113:C      3 else
11114:C      3 begin
11115:C      3 ptr:=anyptr(hex('680000'));
11116:C      3 i:=8;
11117:C      3 while (i<=31) and not found_gator do begin (look for external gator)
11118:C      4 select_code := i;
11119:C      4 try
11120:C      5 dummy:=ptr^;
11121:C      5 if (dummy mod 128) = gatorid then begin
11122:C      6 found_gator:=true;
11123:C      6 control_space:=integer(ptr);
11124:C      6 end;
11125:C      5 recover
11126:C      5 if escapecode<>-12 then escape(escapecode);

```

```

11127:S
11128:C      4      ptr:=anyptr(integer(ptr)+65536);
11129:C      4      i:=i+1;
11130:C      4      end;
11131:C      3      if found_gator then
11132:C      4      begin
11133:C      4      stat := anyptr(control_space + 16384);
11134:C      4      frame_buffer := ((stat^ mod 16)*1048576;
11135:C      4      int_ext_gator := 2;
11136:C      4      end;
11137:C      3      end;
11138:S
11139:S
11140:S      (if found_gator then begin
11141:C      4      try
11142:C      4      has_color:=false; assume no color
11143:C      4      ptr:=anyptr(control_space+hex('410E')); read colormap id
11144:C      4      dummy:=ptr^;
11145:C      4      has_color:=true; set true if color there
11146:C      4      recover
11147:C      4      if escapecode<>-12 then escape(escapecode);
11148:C      2 end;
11149:S
11150:S
11151:S
11152:S
11153:D      1 procedure setupraster ( gcb : graphics_control_block_ptr );
11154:S
11155:D      2 var
11156:D      2 graphics_base ['GRAPHICSBASE'] : anyptr;
11157:D -4 device_work_area : raster_device_rec_ptr;
11158:D -6 cnt : gle_shortint;
11159:D -10 2 address : integer;
11160:D -14 2 control : integer;
11161:D -1S 2 knob_echo_gcb : boolean;
11162:D -20 2 g_ptr : ^shortint;
11163:D -22 2 g_dummy : shortint;
11164:D -23 2 graphics_bd : boolean;
11165:D -23 2 graphics_tate ['GRAPHICSFLAG'] : boolean;
11166:S
11167:D      2 procedure dummy1 ( anyvar iocb_ptr, data_ptr : anyptr );
11168:C      3 begin
11169:C      3 end;
11170:S
11171:D      2 procedure expand_screen;
11172:S
11173:C      3 begin
11174:C      3 with gcb^ do
11175:C      4 begin
11176:C      4 info3 := 0;
11177:C      4 if (int_ext_gator = 1) or (int_ext_gator = 2) then
11178:C      5 begin
11179:C      5 reduced_screen := false;
11180:C      5 info3 := 1; (1=expand; 0=leave reduced)
11181:C      5 end; (send on to expand screen)
11182:C      4 end;
11183:C      3 if (currentcrt = bitmaptype) and (int_ext_gator =1) then
11184:C      4 begin
11185:C      4 save_crtllhook := crtllhook;
11186:C      4 crtllhook := dummycrtll;

```



```

11187:C      4      (gator_clear (gcb));
11188:C      4      took_type_ahead := true;
11189:C      4      end;
11190:C      3 end;
11191:S
11192:D      2      procedure ck_for_graphics_board;
11193:S
11194:C      3      begin
11195:C      3          graphics_bd := true;
11196:C      3          if graphicstate then g_ptr := anyptr(hex('530000'))
11197:C      4              else g_ptr := anyptr(hex('538000'));
11198:C      3          try
11199:C      4              g_dummy := g_ptr^;
11200:C      4              recover
11201:C      4              begin
11202:C      4                  if escapecode <> -12 then escape(escapecode)
11203:C      5                      else graphics_bd := false;
11204:C      4                  end;
11205:C      3          end;
11206:S
11207:S
11208:S
11209:S
11210:D      2      procedure setup_internal;
11211:S
11212:D      3      procedure toggle_graphics;
11213:D      4          var gon [5439488..530000 HEX] : shortint;
11214:D      4          goff [5472256..538000 HEX] : shortint;
11215:D      4          g_on36c [hex('51FFFC')]: shortint;
11216:D      4          gbase['GRAPHICSBASE'] : ^shortint;
11217:C      4          begin
11218:C      4              if gcb^.info1 = m9836c then begin
11219:C      5                  if graphicstate then g_on36c:=1
11220:C      6                      else g_on36c:=0;
11221:C      5                  gbase:=anyptr(hex('520000'));
11222:C      4                  end
11223:C      4                  else begin
11224:C      5                      if graphicstate then gbase := addr(gon)
11225:C      6                          else gbase := addr(goff);
11226:C      5                      gbase^ := gbase^;
11227:C      4                  end;
11228:C      4              end;
11229:S
11230:C      3      begin
11231:C      3          with gcb^ do
11232:C      4              begin
11233:C      4                  graphicstate := true;
11234:C      4                  info1 := return_machine_type;
11235:C      4                  toggle_graphics;
11236:C      4                  info_ptr1 := addr(graphics_base);
11237:C      4                  info_ptr2 := anyptr(0);
11238:C      4                  if info1 = m9836c then
11239:C      5                      begin
11240:C      6                          info2 := hex('51fffd');
11241:C      6                          info3 := hex('51fb00');
11242:C      5                      end;
11243:C      4                  end;
11244:C      3          end;
11245:S
11246:S

```

```

11247:D      2      procedure set_gator_vals;
11248:S
11249:C      3      begin
11250:C      3          with gcb^ do
11251:C      4              begin
11252:C      4                  (if has_color then
11253:C      4                      info1 := mgator_c)
11254:C      4                  (else
11255:C      4                      info1 := m9837a; {display type}
11256:C      4                      info2 := control_space; {top of control space}
11257:C      4                      info3 := 0; {by default dont expand ! BJS 5-29-84}
11258:C      4                      info_ptr1 := addr(frame_buffer); {start of control space}
11259:C      4                      info_ptr2 := anyptr(0);
11260:C      4                  end;
11261:C      3              end;
11262:S
11263:S
11264:S
11265:D      2      begin
11266:C      3          with gcb^ do
11267:C      4              if spooling = 0 then
11268:C      5                  try
11269:C      5                      gatorcrttype(found_gator, has_color,
11270:C      6                          frame_buffer, select_code,int_ext_gator);
11271:C      5                      ck_for_graphics_board;
11272:C      5                      secondary := false;
11273:C      5                      moon := false;
11274:C      5                      reduced_screen := true;
11275:C      5                      took_type_ahead := false;
11276:C      5                      control := info1; { control passed in info1 }
11277:C      5                      knob_echo_gcb := info2 = 1; { GCB for knob echos }
11278:C      5                      io_write := dummy1;
11279:C      5                      io_term := termraster;
11280:C      5                      address := g1e_read_integer(device_info_char_count,device_info,cnt);
11281:C      5                      device_work_area := addr(raster_device_rec_space);
11282:C      5                      dev_dep_stuff := device_work_area;
11283:S
11284:C      5                      if address = 3 then {indicates primary}
11285:C      6                          begin {display}
11286:C      6                              if ((currentcrt = alphaspace) or (currentcrt = nocrt))
11287:C      7                                  and (graphics_bd) then setup_internal
11288:C      8                                  else if ((currentcrt = bitmaptype) and (int_ext_gator = 1))
11289:C      9                                      or ((not graphics_bd) and (int_ext_gator = 1)) then
11290:C      6                                      set_gator_vals;
11291:C      6                              end
11292:C      5                      else if (address = 6) then {indicates secondary }
11293:C      6                          begin {display}
11294:C      6                              secondary := true;
11295:C      6                              if ((currentcrt = alphaspace) or (currentcrt = nocrt)
11296:C      7                                  and (int_ext_gator <> 0)) then
11297:C      8                                  set_gator_vals
11298:C      8                                  else
11299:C      8                                      if (graphics_bd) then setup_internal;
11300:C      8                                  end
11301:C      5                      if (currentcrt = bitmaptype) then {console = gator so}
11302:C      6                          begin {set secondary to }
11303:C      6                              if (graphics_bd) then setup_internal {small screen}
11304:C      7                                  else {if fails set second.}
11305:C      8                                  end
11306:C      5                      end

```



```

11307:C      9      if (int_ext_gator = 1) then      (to gator.)
11308:C      10      set_gator_vals;
11309:C      8      end;
11310:C      7      end;
11311:C      7      else      ( must be moonunit or gator )
11312:C      7      begin
11313:C      7      if (address < minrealisc) or (address > maxrealisc) then escape(1);
11314:C      7      if (address = select_code) and (int_ext_gator = 2) then
11315:C      8      set_gator_vals
11316:C      8      else
11317:C      8      begin
11318:C      8      info3 := control div 256; { get monitor type information (part of control) }
11319:C      8      if (info3 > 6) or (info3 < 1) then info3 := 1;
11320:C      8      moon := true; or
11321:C      8      info1 := m98627a; { set display type to 98627A }
11322:C      8      info2 := address * 65536 + 6291456; { i/o card address }
11323:C      8      hp98627a_address := anyptr(info2 + hex('8000')); { first plane adr }
11324:C      8      info_ptr1 := addr(hp98627a_address);
11325:C      8      info_ptr2 := anyptr(0);
11326:C      8      end;
11327:C      7      end;
11328:S      5
11329:C      5      {control set}
11330:C      5      if (odd(control DIV 256)) and (not moon) then
11331:C      6      if
11332:C      7      ((currentcrt = blitmaptype) and (int_ext_gator = 1) and { is bitmap, primary, }
11333:C      7      (not secondary)) or { gator there }
11334:C      7      or
11335:C      7      (((currentcrt = alphantype) or (currentcrt = nocrt)) and { is alpha/none, gr bd, }
11336:C      7      (graphics_bd) and (secondary) and (int_ext_gator = 1)) { second, gator there }
11337:C      7      or
11338:C      7      (address > 6) and (int_ext_gator = 2) { ext. gator there }
11339:C      7      then expand_screen;
11340:S      5
11341:C      5      {control not set, but gator is not console}
11342:C      5      if (((currentcrt = alphantype) or (currentcrt = nocrt)) and
11343:C      6      (int_ext_gator <> 0) and
11344:C      6      (secondary))
11345:C      6      or
11346:C      6      (((currentcrt = alphantype) or (currentcrt = nocrt)) and
11347:C      6      (not graphics_bd) and (int_ext_gator <> 0))
11348:C      6      then expand_screen;
11349:S      5
11350:S      5      gle_init_raster_output (gcb);
11351:C      5
11352:S      5
11353:S      5      if (error_return = 0) and (not knob_echo_gcb) then dgl_raster_init(control);
11354:C      5
11355:S      5      {if error_return <> 0 then
11356:C      5      dispose(device_work_area);} { clean up }
11357:C      5
11358:C      5      recover
11359:S      5      { ignore all escapes (except stop key), user may look at
11360:C      5      escapecode to determine error }
11361:C      5      if escapecode = -20 then escape(-20)
11362:C      6      else error_return := 1
11363:C      6      else
11364:C      4      error_return := 1; { raster devices may not be spooled }
11365:C      2      end;
11366:S      2

```

```

11367:D      1      procedure termhpgl ( anyvar iocb_ptr : anyptr );
11368:S      2
11369:D      2      var
11370:D      -4      iocb_ptr_file : file_iocb_ptr;
11371:D      -8      iocb_ptr_hpib : hpib_iocb_ptr;
11372:D      -12      buf : ascii_buffer_ptr;
11373:D      -16      device_work_area : hpgl_device_rec_ptr;
11374:D      -20      save_ioreresult : integer; { | fix clobbering ioreresult -- 12/83}
11375:S      2
11376:C      2      begin
11377:C      2      with gle_gcb^ do
11378:C      3      begin
11379:C      3      if spooling <> 0 then
11380:C      4      begin
11381:C      4      save_ioreresult := ioreresult; { | ioreresult problem fix 12/83--8DS }
11382:C      4      file_term(iocb_ptr); { perform io term then release mem }
11383:C      4      ioreresult := save_ioreresult; { | ioreresult problem fix 12/83--8DS }
11384:C      4      iocb_ptr_file := iocb;
11385:C      4      {dispose(iocb_ptr_file);}
11386:C      4      end
11387:C      4      else
11388:C      4      begin
11389:C      4      hpib_term(iocb_ptr); { perform io term then release mem }
11390:C      4      iocb_ptr_hpib := iocb;
11391:C      4      {dispose(iocb_ptr_hpib);}
11392:C      4      end;
11393:C      3      buf := device_buf;
11394:C      3      device_work_area := dev_dep_stuff;
11395:C      3      {dispose(buf); dispose(device_work_area);}
11396:C      3      end;
11397:C      2      end;
11398:S      2
11399:D      1      procedure setuphpgl ( gcb : graphics_control_block_ptr );
11400:S      2
11401:D      2      var
11402:D      -4      iocb_ptr_file : file_iocb_ptr;
11403:D      -8      iocb_ptr_hpib : hpib_iocb_ptr;
11404:D      -12      buf : ascii_buffer_ptr;
11405:D      -16      device_work_area : hpgl_device_rec_ptr;
11406:D      -18      cnt : gle_shortint;
11407:D      -22      address : integer;
11408:D      -23      address_found : boolean;
11409:D      -28      control : integer;
11410:D      -32      save_ioreresult : integer; { | fix clobbering ioreresult -- 12/83}
11411:D      -36      save : integer;
11412:S      2
11413:C      2      begin
11414:C      2      with gcb^ do
11415:C      3      begin
11416:C      3      control := info1; { control passed in info1 }
11417:C      3      address_found := false;
11418:C      3      try
11419:C      4      address := gle_read_integer(device_info_char_count, device_info, cnt);
11420:C      4      address_found := true;
11421:C      4      recover
11422:C      4      if escapecode <> -8 { value range error } then escape(escapecode);
11423:S      3
11424:C      3      buf := addr(ascii_buffer_space);
11425:C      3      device_buf := buf;
11426:C      3      device_work_area := addr(hpgl_device_rec_space);

```



```

11427:C      3      dev_dep_stuff := device_work_area;
11428:S
11429:C      3      if spooling = 1 then
11430:C      4      begin
11431:C      4          iocb_ptr_file := addr(file_iocb_space);
11432:C      4          iocb := iocb_ptr_file;
11433:C      4          io_write := file_write;
11434:C      4          io_term := termhpgl;
11435:C      4          io_inq_timeout := file_inq_timeout;
11436:C      4          io_set_timeout := file_set_timeout;
11437:C      4          with iocb_ptr_file do
11438:C      5              begin
11439:C      5                  file_name := device_info;
11440:C      5                  name_size := device_info_char_count;
11441:C      5                  try
11442:C      6                      lock_on_close := 0;          ( do not save file by default )
11443:C      6                      file_init ( iocb_ptr_file );
11444:C      6                      gle_init_hpgl_output (gcb);
11445:C      6                      if error_return = 0 then
11446:C      7                          begin
11447:C      7                              dgl_hpgl_init(control);
11448:C      7                              lock_on_close := 1;          ( save file )
11449:C      7                          end
11450:C      7                      else
11451:C      7                          begin
11452:C      7                              save_ioreult := ioreult;          (| ioreult fix 12/83--BDS)
11453:C      7                              file_term ( iocb_ptr_file );
11454:C      7                              ioreult := save_ioreult;          (| ioreult fix 12/83--BDS)
11455:C      7                          end;
11456:C      6                      recover
11457:C      6                      if escapecode <> -10 then escape(escapecode)
11458:C      7                      else error_return := 1;
11459:C      5                  end;
11460:S
11461:C      4                  if error_return <> 0 then
11462:C      5                      begin ( clean up )
11463:C      5                          save_ioreult := ioreult;          (| ioreult fix 1/84--BDS)
11464:C      5                          (dispose(iocb_ptr_file); dispose(buf); dispose(device_work_area);)
11465:C      5                          ioreult := save_ioreult;          (| ioreult fix 1/84--BDS)
11466:C      5                      end;
11467:C      4                  end
11468:C      4              else
11469:C      4                  if address_found then
11470:C      5                      begin
11471:C      5                          iocb_ptr_hpib := addr(hpib_iocb_space);
11472:C      5                          iocb := iocb_ptr_hpib;
11473:C      5                          io_write := hpib_write;
11474:C      5                          io_read := hpib_read;
11475:C      5                          io_term := termhpgl;
11476:C      5                          io_inq_timeout := hpib_inq_timeout;
11477:C      5                          io_set_timeout := hpib_set_timeout;
11478:C      5                          with iocb_ptr_hpib do
11479:C      6                              begin
11480:C      6                                  device_addr := device_info;
11481:C      6                                  name_size := device_info_char_count;
11482:C      6                                  end;
11483:C      5                          hpib_init ( iocb_ptr_hpib );
11484:C      5                          if iocb_ptr_hpib.error = 0 then
11485:C      6                              begin
11486:C      6                                  gle_init_hpgl_output (gcb);

```

```

11487:C      6          if error_return = 0 then dgl_hpgl_init(control)
11488:C      7          { if error then clean up hpib bus (2.1 bug fix) }
11489:C      7          else
11490:C      6              end
11491:C      6          else error_return := 1;
11492:C      6          if error_return <> 0 then
11493:C      6              begin ( clean up )
11494:C      6                  (dispose(iocb_ptr_hpib); dispose(buf); dispose(device_work_area);)
11495:C      6              end;
11496:C      5          end
11497:C      5          else
11498:C      5              error_return := 1;
11499:C      3          end;
11500:C      2      end;
11501:S
11502:D      1      procedure configure_gle ( gcb : graphics_control_block_ptr );
11503:S
11504:D      2      VAR
11505:D      -4      2      save : integer;
11506:S
11507:C      2      begin
11508:C      2          with gcb do
11509:C      3              begin
11510:C      3                  setupraster ( gcb );
11511:C      3                  if error_return <> 0 then setuphpgl ( gcb );
11512:C      3                  end;
11513:C      2      end;
11514:S
11515:C      1      end. ( of module )
11516:S
11517:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

DGL_HPGL

Description

DGL_HPGL provides device-dependent initialization of HPGL devices, input/output escape functions, and HPGL plotter setup utilities.

Requirements

DGL_TYPES, DGL_TYPES, DGL_VARS, GLE_TYPES, GLE_GEN, GLE_UTLS, SYSGLOBALS, and LOADER.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1 D      0
2 DD     0 DGL device dependent init routine
3 DD     0
4 DD     0 Module = DGL_HPGL
5 DD     0 Programmer = BJS
6 DD     0 Date = 10- 5-82
7 DD     0
8 DD     0 Purpose: To provide device dependent initialization for HPGL devices.
9 DD     0
10 DD    0 Rev history
11 DD    0 Created - 1 - 5-82 BJS
12 DD    0 Modified - 4 -03-84 BJS Changes dynamic allocations to globals for 3.0
13 DD    0 Added identifiers for 7586,7550,7475
14 DD    0
15 DD    0 (c) Copyright Hewlett-Packard Company, 1983.
16 DD    0 All rights are reserved. Copying or other
17 DD    0 reproduction of this program except for archival
18 DD    0 purposes is prohibited without the prior
19 DD    0 written consent of Hewlett-Packard Company.
20 DD    0
21 DD    0
22 DD    0
23 DD    0
24 DD    0
25 DD    0
26 DD    0
27 DD    0
28 DD    0
29 DD    0
30 DD    0 HEWLETT-PACKARD COMPANY
31 DD    0 Fort Collins, Colorado
32 DD    0
33 DD    0 $search 'GLE_LIB',
34 DD    0 'TYPES'
35 DD    0 'DGL_VARS'$
36 DD    0 $modals$
37 DD    0 $include 'OPTIONS'$ { ***** COMPILER OPTIONS ***** }
38 DD    0
39 DD    0 { This include file specifies range checking, debug and other compiler
40 DD    0 options for the graphics library }
41 DD    0 $debug OFF$
42 DD    0 $range OFF$
43 DD    0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
44 DD    0 $FLOAT_HDW TEST$
45 DD    0
46 DD    0
47 DD    0
48 DD    0
49 DD    0 $include 'OPTIONS'$ { ***** COMPILER OPTIONS ***** }
50 DD    0 $linenum 17000$
51 DD    0
52 DD    0 module DGL_HPGL;
53 DD    0
54 DD    0 1 export
55 DD    0
56 DD    0 1 procedure dgl_hppl_init(control : integer);
57 DD    0
58 DD    0 1 implement
59 DD    0
60 DD    0
61 DD    0
62 DD    0
63 DD    0
64 DD    0
65 DD    0
66 DD    0
67 DD    0
68 DD    0
69 DD    0
70 DD    0
71 DD    0
72 DD    0
73 DD    0
74 DD    0
75 DD    0
76 DD    0
77 DD    0
78 DD    0
79 DD    0
80 DD    0
81 DD    0
82 DD    0
83 DD    0
84 DD    0
85 DD    0
86 DD    0
87 DD    0
88 DD    0
89 DD    0
90 DD    0
91 DD    0
92 DD    0
93 DD    0
94 DD    0
95 DD    0
96 DD    0
97 DD    0
98 DD    0
99 DD    0
100 DD   0

```

```

17010:D 1 import dgl_types,
17011:D 1 dgl_vars,
17012:D 1 gle_gen,
17013:D 1 gle_util,
17014:D 1 asm;
17015:S
17016:D 1 procedure hppl_input_esc ( opcode : integer;
17017:D 2 isize : integer;
17018:D 2 rsize : integer;
17019:D 2 anyvar ilist : gint_list;
17020:D 2 anyvar rlist : greal_list;
17021:D 2 var ierr : integer );
17022:S
17023:S
17024:D 2 { Purpose : To perform an input escape function }
17025:S
17026:D 2 var
17027:D -42 s : string[40];
17028:D -42 cnt : integer;
17029:D -47 c : char;
17030:S
17031:D 2 begin
17032:D 2 with gle_gcb do
17033:D 3 begin
17034:D 3 if spooling = 1 then ierr := 4;
17035:D 3 if (gle_match(4,addr(display_name),4,addr('7580')) or
17036:D 4 gle_match(4,addr(display_name),4,addr('7550')) or
17037:D 4 gle_match(4,addr(display_name),4,addr('7475')) or
17038:D 4 gle_match(4,addr(display_name),4,addr('7090')) or
17039:D 4 gle_match(4,addr(display_name),4,addr('7585')) or
17040:D 4 gle_match(4,addr(display_name),4,addr('7586')))) and
17041:D 4 (opcode = 2050) then
17042:D 4 begin
17043:D 4 if (ierr = 0) then
17044:D 5 begin
17045:D 5 info_ptr1 := addr('0T');
17046:D 5 infoI := 2;
17047:D 5 gle_output_escape( gle_gcb );
17048:D 5 gle_flush_buffer (gle_gcb);
17049:D 5 info_ptr1 := addr(s[1]);
17050:D 5 gle_output_escape( gle_gcb );
17051:D 5 setstrlen(s,infoI);
17052:D 5 strread(s,1,cnt,ilist[1],c,ilist[2]);
17053:D 5 end;
17054:D 4 end
17055:D 4 else
17056:D 4 ierr := 1;
17057:D 3 end;
17058:D 2 end; { input_esc }
17059:S
17060:D 1 procedure hppl_output_esc ( opcode : integer;
17061:D 2 isize : integer;
17062:D 2

```



```

17063:0      2      rsize : integer;
17064:0      2      anyvar ilist : gint_list;
17065:0      2      anyvar rlist : greal_list;
17066:0      2      var ierr : integer );
17067:S
17068:0      2 ( Purpose : To perform an output escape function )
17069:S
17070:0      2 var
17071:0      -22 2 s : string[20];
17072:0      -26 2 cnt : integer;
17073:S
17074:0      2 procedure set_auto_pen;
17075:S
17076:C      3 begin
17077:C      3   strwrite(s,1,cnt,'AP',ilist[1]:1);
17078:C      3 end;
17079:S
17080:0      2 procedure control_cutter;
17081:S
17082:C      3 begin
17083:C      3   s := 'EC';
17084:C      3   if ilist[1] = 0 then
17085:C      4     begin setstrlen(s,3); s[3] := 'C'; end;
17086:C      3 end;
17087:S
17088:0      2 procedure advance_page;
17089:S
17090:C      3 begin
17091:C      3   if ilist[1] = 0 then s := 'AH'
17092:C      4   else s := 'AF';
17093:C      3 end;
17094:S
17095:0      2 procedure set_velocity;
17096:S
17097:C      3 begin
17098:C      3   strwrite(s,1,cnt,'VS',ilist[1]:1);
17099:C      3   if (ilist[2] > 0) and (ilist[2] < gle_gcb^.gamut) then
17100:C      4     strwrite(s,cnt,cnt,',',ilist[2]:1);
17101:C      3 end;
17102:S
17103:0      2 procedure set_force;
17104:S
17105:C      3 begin
17106:C      3   strwrite(s,1,cnt,'FS',ilist[1]:1);
17107:C      3   if (ilist[2] > 0) and (ilist[2] < gle_gcb^.gamut) then
17108:C      4     strwrite(s,cnt,cnt,',',ilist[2]:1);
17109:C      3 end;
17110:S
17111:0      2 procedure set_acceleration;
17112:S
17113:C      3 begin
17114:C      3   strwrite(s,1,cnt,'AS',ilist[1]:1);
17115:C      3   if (ilist[2] > 0) and (ilist[2] < gle_gcb^.gamut) then
17116:C      4     strwrite(s,cnt,cnt,',',ilist[2]:1);
17117:C      3 end;
17118:S
17119:0      2 begin
17120:C      2   with gle_gcb^ do
17121:C      3   begin
17122:C      3     s := '';

```

```

***WARNING: (line17123): 'ADDR' of a constant may not be supported on other implementations
17123:C      3   if gle_match(4,addr(display_name),4,addr('9872')) then
17124:C      4     begin
17125:C      4       if (opcode = 1052) then
17126:C      5         begin if (ierr = 0) then control_cutter; end
17127:C      5       else
17128:C      5         if (opcode = 1053) then
17129:C      6           begin if (ierr = 0) then advance_page; end
17130:C      6         else
17131:C      6           if (opcode = 2050) then
17132:C      7             begin if (ierr = 0) then set_velocity; end
17133:C      7           else
17134:C      7             ierr := 1;
17135:C      4         end
17136:C      4       else
***WARNING: (line17137): 'ADDR' of a constant may not be supported on other implementations
17137:C      4       if ((gle_match(4,addr(display_name),4,addr('7470')) or
***WARNING: (line17138): 'ADDR' of a constant may not be supported on other implementations
17138:C      5       (gle_match(4,addr(display_name),4,addr('7440')))) then
17139:C      5         begin
17140:C      5           if (opcode = 2050) then
17141:C      6             begin if (ierr = 0) then set_velocity; end
17142:C      6           else
17143:C      6             ierr := 1;
17144:C      5         end
17145:C      5       else
***WARNING: (line17146): 'ADDR' of a constant may not be supported on other implementations
17146:C      5       if (gle_match(4,addr(display_name),4,addr('7580')) or
***WARNING: (line17147): 'ADDR' of a constant may not be supported on other implementations
17147:C      6       gle_match(4,addr(display_name),4,addr('7550')) or
***WARNING: (line17148): 'ADDR' of a constant may not be supported on other implementations
17148:C      6       gle_match(4,addr(display_name),4,addr('7475')) or
***WARNING: (line17149): 'ADDR' of a constant may not be supported on other implementations
17149:C      6       gle_match(4,addr(display_name),4,addr('7090')) or
***WARNING: (line17150): 'ADDR' of a constant may not be supported on other implementations
17150:C      6       gle_match(4,addr(display_name),4,addr('7586')) or
***WARNING: (line17151): 'ADDR' of a constant may not be supported on other implementations
17151:C      6       gle_match(4,addr(display_name),4,addr('7586')))) then
17152:C      6         begin
17153:C      6           if (opcode = 1052) then
17154:C      7             begin if (ierr = 0) then set_auto_pen; end
17155:C      7           else
17156:C      7             if ((opcode = 1053) and
***WARNING: (line17157): 'ADDR' of a constant may not be supported on other implementations
17157:C      8             ((gle_match(4,addr(display_name),4,addr('7586')) or
***WARNING: (line17158): 'ADDR' of a constant may not be supported on other implementations
17158:C      8             (gle_match(4,addr(display_name),4,addr('7550')))))
17159:C      8             then
17160:C      8               begin if (ierr = 0) then advance_page; end
17161:C      8             else
17162:C      8               if (opcode = 2050) then
17163:C      9                 begin if (ierr = 0) then set_velocity; end
17164:C      9               else
17165:C      9               if (opcode = 2051) then
17166:C      10                begin if (ierr = 0) then set_force; end
17167:C      10              else
17168:C      10              if (opcode = 2052) then
17169:C      11                begin if (ierr = 0) then set_acceleration; end
17170:C      11              else
17171:C      11                ierr := 1;

```



```

1717:C      6      end
1717:C      6      else
1717:C      6      ierr := 1;
1717:C      6      if ierr = 0 then
1717:C      4      begin
1717:C      4      info_ptr1 := addr(s[1]);
1717:C      4      info1 := strlen(s);
1717:C      4      gle_output_escaped(gle_gcb);
1718:C      4      end;
1718:C      3      end;
1718:C      2      end; { output_esc }
1718:C      1      procedure hpgl_linestyle ( index : integer);
1718:C      2      { Purpose: To set the linestyle that primitives are drawn with }
1718:C      2      type
1718:C      2      ls_map_def = packed array [1..13] of gbyte;
1718:C      2      const
1718:C      2      ls_map = ls_map_def [0,2,3,4,5,6,1,2,3,4,5,6,1];
1718:C      2      begin
1718:C      2      with gle_gcb^ do
1718:C      2      begin
1718:C      2      info1 := ls_map[index];           { map DGL to GLE def }
1718:C      2      info2 := 4;                     { repeat rate 4% }
1718:C      2      if (index > 7) then info3 := 1
1718:C      2      else info3 := 0;                 { linestyle mode }
1718:C      2      info4 := 0;
1718:C      2      gle_linestyle ( gle_gcb );
1718:C      2      end;
1718:C      2      end; { hpgl_linestyle }
1718:C      1      procedure hpgl_color ( index : integer );
1718:C      2      begin
1718:C      2      gle_gcb^.info1 := index;
1718:C      2      gle_index_color ( gle_gcb );
1718:C      2      end;
1718:C      1      procedure hpgl_color_table ( index : integer;
1718:C      2      parm1 : real;
1718:C      2      parm2 : real;
1718:C      2      parm3 : real);
1718:C      2      begin
1718:C      2      end;
1718:C      1      procedure dgl_hpgl_init(control : integer);
1718:C      2      type
1718:C      2      default_poly_table_def = array[1..16] of poly_entry_def;
1718:C      2      const
1718:C      2      default_poly_table = default_poly_table_def [
1718:C      2      poly_entry_def [ density : 0.0, orient : 0.0, edge : true ], { 1 }
1718:C      2      poly_entry_def [ density : 0.125, orient : 90.0, edge : true ], { 2 }
1718:C      2      poly_entry_def [ density : 0.125, orient : 0.0, edge : true ], { 3 }
1718:C      2      poly_entry_def [ density : 0.125, orient : 0.0, edge : true ], { 4 }
1718:C      2      poly_entry_def [ density : -0.125, orient : 0.0, edge : true ], { 5 }
1718:C      2      poly_entry_def [ density : 0.125, orient : 45.0, edge : true ], { 6 }
1718:C      2      poly_entry_def [ density : -0.125, orient : -45.0, edge : true ], { 7 }
1718:C      2      poly_entry_def [ density : 0.25, orient : 90.0, edge : true ], { 8 }
1718:C      2      poly_entry_def [ density : 0.25, orient : 0.0, edge : true ], { 9 }
1718:C      2      poly_entry_def [ density : -0.25, orient : 0.0, edge : true ], { 10 }
1718:C      2      poly_entry_def [ density : 0.25, orient : 45.0, edge : true ], { 11 }
1718:C      2      poly_entry_def [ density : 0.25, orient : -45.0, edge : true ], { 12 }
1718:C      2      poly_entry_def [ density : -0.25, orient : 45.0, edge : true ], { 13 }
1718:C      2      poly_entry_def [ density : -0.5, orient : 0.0, edge : true ], { 14 }
1718:C      2      poly_entry_def [ density : 1.0, orient : 0.0, edge : false ], { 15 }
1718:C      2      poly_entry_def [ density : 1.0, orient : 0.0, edge : true ], { 16 }
1718:C      2      type
1718:C      2      control_def = packed record
1718:C      2      case gshortint of
1718:C      2      0 : (whole : gshortint);
1718:C      2      1 : (part : packed record
1718:C      2      b15,b14,b13,b12,
1718:C      2      b11,b10,b9, b8,
1718:C      2      clr_inhibit,b6,b5,b4,
1718:C      2      b3,b2,b1,b0 : boolean;
1718:C      2      end);
1718:C      2      end;
1718:C      2      var
1718:C      2      temp_control : control_def;
1718:C      2      i : integer;
1718:C      2      begin
1718:C      2      with gcb^ do
1718:C      2      begin
1718:C      2      disp_just := lowerleft;
1718:C      2      clipping_support := true;
1718:C      2      retroactive_color_support := false;
1718:C      2      retroactive_polygon_support := false;
1718:C      2      maximum_polygon_vertices := 0; { no hardware support }
1718:C      2      if gle_gcb^.vect_linestyles <= 0 then number_dgl_linestyles := 13
1718:C      2      else number_dgl_linestyles := 7;
1718:C      2      number_markers := 19;
1718:C      2      proc_output_esc := hpgl_output_esc;
1718:C      2      proc_input_esc := hpgl_input_esc;
1718:C      2      proc_linestyle := hpgl_linestyle;
1718:C      2      proc_color := hpgl_color;
1718:C      2      proc_color_table := hpgl_color_table;
1718:C      2      color_table_size := 0;
1718:C      2      { allocate polygon table space }
1718:C      2      number_polygon_styles := 16;
1718:C      2      (newbytes(poly_table_ptr,number_polygon_styles * 18));
1718:C      2      poly_table_ptr := addr(poly_table_def_space);
1718:C      2      for i := 1 to poly_table_size do
1718:C      2      poly_table_ptr[i] := default_poly_table[i];
1718:C      2      display_echo_mult := 8;
1718:C      2      temp_control.whole := control;
1718:C      2      if not temp_control.part.clr_inhibit then
1718:C      2      with gle_gcb^ do

```

```

1723:C      2      poly_entry_def [ density : 0.125, orient : -45.0, edge : true ], { 6 }
1723:C      2      poly_entry_def [ density : -0.125, orient : 45.0, edge : true ], { 7 }
1723:C      2      poly_entry_def [ density : 0.25, orient : 90.0, edge : true ], { 8 }
1723:C      2      poly_entry_def [ density : 0.25, orient : 0.0, edge : true ], { 9 }
1723:C      2      poly_entry_def [ density : -0.25, orient : 0.0, edge : true ], { 10 }
1723:C      2      poly_entry_def [ density : 0.25, orient : 45.0, edge : true ], { 11 }
1723:C      2      poly_entry_def [ density : 0.25, orient : -45.0, edge : true ], { 12 }
1723:C      2      poly_entry_def [ density : -0.25, orient : 45.0, edge : true ], { 13 }
1723:C      2      poly_entry_def [ density : -0.5, orient : 0.0, edge : true ], { 14 }
1723:C      2      poly_entry_def [ density : 1.0, orient : 0.0, edge : false ], { 15 }
1723:C      2      poly_entry_def [ density : 1.0, orient : 0.0, edge : true ], { 16 }
1723:C      2      type
1723:C      2      control_def = packed record
1723:C      2      case gshortint of
1723:C      2      0 : (whole : gshortint);
1723:C      2      1 : (part : packed record
1723:C      2      b15,b14,b13,b12,
1723:C      2      b11,b10,b9, b8,
1723:C      2      clr_inhibit,b6,b5,b4,
1723:C      2      b3,b2,b1,b0 : boolean;
1723:C      2      end);
1723:C      2      end;
1723:C      2      var
1723:C      2      temp_control : control_def;
1723:C      2      i : integer;
1723:C      2      begin
1723:C      2      with gcb^ do
1723:C      2      begin
1723:C      2      disp_just := lowerleft;
1723:C      2      clipping_support := true;
1723:C      2      retroactive_color_support := false;
1723:C      2      retroactive_polygon_support := false;
1723:C      2      maximum_polygon_vertices := 0; { no hardware support }
1723:C      2      if gle_gcb^.vect_linestyles <= 0 then number_dgl_linestyles := 13
1723:C      2      else number_dgl_linestyles := 7;
1723:C      2      number_markers := 19;
1723:C      2      proc_output_esc := hpgl_output_esc;
1723:C      2      proc_input_esc := hpgl_input_esc;
1723:C      2      proc_linestyle := hpgl_linestyle;
1723:C      2      proc_color := hpgl_color;
1723:C      2      proc_color_table := hpgl_color_table;
1723:C      2      color_table_size := 0;
1723:C      2      { allocate polygon table space }
1723:C      2      number_polygon_styles := 16;
1723:C      2      (newbytes(poly_table_ptr,number_polygon_styles * 18));
1723:C      2      poly_table_ptr := addr(poly_table_def_space);
1723:C      2      for i := 1 to poly_table_size do
1723:C      2      poly_table_ptr[i] := default_poly_table[i];
1723:C      2      display_echo_mult := 8;
1723:C      2      temp_control.whole := control;
1723:C      2      if not temp_control.part.clr_inhibit then
1723:C      2      with gle_gcb^ do

```



```
17292:C      S      begin
17293:C      S      info1 := -1;
17294:C      S      info2 := dgl_background_index; ( clear all planes )
17295:C      S      gle_clear ( gle_gcb );
17296:C      S      end;
17297:C      3      end;
17298:C      2      end;
17299:S      1      end. ( dgl_hppl )
17300:C
17301:S
```

No errors. 18 warnings.

***** Nonstandard language features enabled *****


```

1:D      0 { DGL device dependent init routine }
2:D      0 { }
3:D      0 { }
4:D      0 { Module = DGL_HPGLI }
5:D      0 { Programmer = BJS }
6:D      0 { Date = 2-10-83 }
7:D      0 { }
8:D      0 { Purpose: To provide device dependent initialization for the knob. }
9:D      0 { }
10:D     0 { Rev history }
11:D     0 { Created - 2-10-82 BJS }
12:D     0 { Modified - XX-XX-XX }
13:S
14:S     ( (c) Copyright Hewlett-Packard Company, 1983.
15:S     All rights are reserved. Copying or other
16:S     reproduction of this program except for archival
17:S     purposes is prohibited without the prior
18:S     written consent of Hewlett-Packard Company.
19:S
20:S
21:S     RESTRICTED RIGHTS LEGEND
22:S
23:S     Use, duplication, or disclosure by the Government
24:S     is subject to restrictions as set forth in
25:S     paragraph (b) (3) (B) of the Rights in Technical
26:S     Data and Computer Software clause in
27:S     DAR 7-104.9(a).
28:S
29:S     HEWLETT-PACKARD COMPANY
30:D     0 Fort Collins, Colorado }
31:S
32:D     0 $search 'GLE_LIB',
33:D     0 'TYPES',
34:D     0 'DGL_VARS',
35:D     0 'GEN',
36:D     0 'DGL_C_OUT'S
37:D     0 $modcall$
38:D     0 $include 'OPTIONS'S { ***** COMPILER OPTIONS ***** }
39:S
40:D     0 { This include file specifies range checking, debug and other compiler
41:D     0 options for the graphics library }
42:D     0 $debug OFF$
43:D     0 $range OFF$
44:D     0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'S
45:D     0 $FLOAT_HDW TEST$
46:S
47:S
48:S
49:S
50:D     0 $include 'OPTIONS'S { ***** COMPILER OPTIONS ***** }
18000:D     0 $linenum 18000$
18001:S
18002:D     0 module DGL_HPGLI;
18003:S
18004:D     1 export
18005:S
18006:D     1 procedure dgl_hppli_init;
18007:S
18008:D     1 implement

```

```

18009:S
18010:D     1 import dgl_types,
18011:D     1 dgl_vars,
18012:D     1 gle_types,
18013:D     1 gle_gen,
18014:D     1 dgl_gen;
18015:D
18016:S
18017:D     1 procedure hpgli_sample_locator( echo : integer;
18018:D     2 var rx,ry : real );
18019:D
18020:D     2 { Purpose : To sample the locator device }
18021:S
18022:D     2 var
18023:D     -8 2 dx, dy : integer;
18024:S
18025:D     2 begin
18026:D     2 with gle_gcbi^ do
18027:D     2 begin
18028:D     3 gle_sample(gle_gcbi);
18029:D     3 convert_lto2(info1,info2,dx,dy);
18030:D     3 convert_dto2(dx,dy,rx,ry);
18031:D     3 info1 := echo;
18032:D     3 gle_input_echo(gle_gcbi); { echo on locator device }
18033:D     3 end;
18034:D     2 end; { sample_locator }
18035:S
18036:D     1 procedure hpgli_await_locator(var echo : integer;
18037:D     2 var button : integer;
18038:D     2 var rx,ry : real );
18039:D
18040:D     2 { Purpose : To activate the locator, and wait for operator termination }
18041:S
18042:D     2 var
18043:D     -1 echoerror : boolean;
18044:D     -10 2 dx,dy : integer;
18045:D     -18 2 last_x,last_y : integer;
18046:S
18047:D     2 begin
18048:D     2 if (echo < 0) or (echo > 8) then echo := 1; { ck echo range }
18049:D     2 { ck for display echo, and display not enabled }
18050:D     2 if (not disp_init) and (echo > 1) then
18051:D     3 begin
18052:D     3 echoerror := true;
18053:D     3 echo := 1;
18054:D     3 end
18055:D     3 else
18056:D     3 echoerror := false;
18057:D     3
18058:D     2 with gcb^,gle_gcbi^ do
18059:D     3 begin
18060:D     3 current_echo_type := echo;
18061:D     3
18062:D     3 info2 := 0;
18063:D     3 gle_start_digitize(gle_gcbi);
18064:D     3
18065:D     3 last_x := -32768;
18066:D     3 last_y := -32768;
18067:S

```



```

18069:C      3      if echo > 1 then
18070:C      4          with gle_gcb^ do
18071:C      5              begin
18072:C      6                  info1 := d_loc_echo_x;
18073:C      7                  info2 := d_loc_echo_y;
18074:C      8                  info3 := 1; { on }
18075:C      9                  gle_cursor ( gle_gcb ; ; { perform first echo at lep }
18076:C      6              end;
18077:S
18078:C      3      repeat
18079:C      4          gle_sample ( gle_gcbi );
18080:C      4          convert_ltod(info1,info2,dx,dy);
18081:C      4          button := info3;
18082:S
18083:C      4          if (dx <> last_x) or (dy <> last_y) then
18084:C      5              begin
18085:C      6                  last_x := dx;
18086:C      7                  last_y := dy;
18087:C      8                  if (echo > 1) and (not disp_eq_loc) then
18088:C      9                      with gle_gcb^ do
18089:C      6                          begin
18090:C      7                              info1 := dx;
18091:C      7                              info2 := dy;
18092:C      7                              info3 := 1;
18093:C      7                              gle_cursor ( gle_gcb );
18094:C      7                          end;
18095:C      5                      end;
18096:C      4          until button = -1;
18097:S
18098:C      3          gle_get_digitize(gle_gcbi);
18099:C      3          button := info3;
18100:S
18101:C      3          convert_ltod(info1,info2,dx,dy);
18102:C      3          convert_dto(dx,dy,rx,ry);
18103:C      3          adjust_return_echo ( rx, ry );
18104:S
18105:C      3          if echo > 1 then
18106:C      4              with gle_gcb^ do
18107:C      5                  begin
18108:C      6                      info3 := 0;
18109:C      7                      gle_cursor(gle_gcb); { remove cursor from screen }
18110:C      5                  end;
18111:S
18112:C      3          if echo = 1 then
18113:C      4              begin
18114:C      5                  info1 := echo;
18115:C      6                  gle_input_echo ( gle_gcbi );
18116:C      4              end;
18117:C      3          end;
18118:S
18119:C      2          if echoerror then error (err_echo_dis_int);
18120:S
18121:C      2      end; { await_locator }
18122:S
18123:C      1      procedure dgl_hpgli_init;
18124:S
18125:C      2      begin
18126:C      2          with gcb^ do
18127:C      3              begin
18128:C      3                  proc_await_locator := hpgli_await_locator;

```

```

18129:C      3          proc_sample_locator := hpgli_sample_locator;
18130:C      3          end;
18131:C      2      end;
18132:S
18133:C      1      end. { dgl_hpgli_init }
18134:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

DGL_INQ

Description

DGL_INQ provides user inquiry code for DGL.

Requirements

DGL_TYPES, DGL_VARS, GLE_TYPES, GLE_RAS_OUT, SYSGLOBALS, SYSDEVS, GLE_ARAS_OUT, GLE_ASCLIP, GLE_STEXT, GLE_ASTEXT, GLE_SCLIP, GLE_ASCLIP, GLE_SMARK, and GLE_AUTL.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:0      0 {
2:0      0 { Pascal work station graphics library }
3:0      0 {
4:0      0 { Module = DGL_INQ
5:0      0 { Programmer = BJS
6:0      0 { Date = 4/13/82
7:0      0 {
8:0      0 { Purpose: Holds all code for user inquiries to the DGL system. }
9:0      0 {
10:0     0 { (c) Copyright Hewlett-Packard Company, 1983.
11:0     0 { All rights are reserved. Copying or other
12:0     0 { reproduction of this program except for archival
13:0     0 { purposes is prohibited without the prior
14:0     0 { written consent of Hewlett-Packard Company.
15:0     0 {
16:0     0 {
17:0     0 {
18:0     0 {
19:0     0 {
20:0     0 {
21:0     0 {
22:0     0 {
23:0     0 {
24:0     0 {
25:0     0 {
26:0     0 {
27:0     0 {
28:0     0 {
29:0     0 {
30:0     0 {
31:0     0 {
32:0     0 {
33:0     0 {
34:0     0 {
35:0     0 {
36:0     0 {
37:0     0 {
38:0     0 {
39:0     0 {
40:0     0 {
41:0     0 {
42:0     0 {
43:0     0 {
44:0     0 {
45:0     0 {
46:0     0 {
47:0     0 {
48:0     0 {
49:0     0 {
50:0     0 {
51:0     0 {
52:0     0 {
53:0     0 {
54:0     0 {
55:0     0 {
56:0     0 {
57:0     0 {
58:0     0 {
59:0     0 {
60:0     0 {
61:0     0 {
62:0     0 {
63:0     0 {
64:0     0 {
65:0     0 {
66:0     0 {
67:0     0 {
68:0     0 {
69:0     0 {
70:0     0 {
71:0     0 {
72:0     0 {
73:0     0 {
74:0     0 {
75:0     0 {
76:0     0 {
77:0     0 {
78:0     0 {
79:0     0 {
80:0     0 {
81:0     0 {
82:0     0 {
83:0     0 {
84:0     0 {
85:0     0 {
86:0     0 {
87:0     0 {
88:0     0 {
89:0     0 {
90:0     0 {
91:0     0 {
92:0     0 {
93:0     0 {
94:0     0 {
95:0     0 {
96:0     0 {
97:0     0 {
98:0     0 {
99:0     0 {
100:0    0 {
101:0    0 {
102:0    0 {
103:0    0 {
104:0    0 {
105:0    0 {
106:0    0 {
107:0    0 {
108:0    0 {
109:0    0 {
110:0    0 {
111:0    0 {
112:0    0 {
113:0    0 {
114:0    0 {
115:0    0 {
116:0    0 {
117:0    0 {
118:0    0 {
119:0    0 {
120:0    0 {
121:0    0 {
122:0    0 {
123:0    0 {
124:0    0 {
125:0    0 {
126:0    0 {
127:0    0 {
128:0    0 {
129:0    0 {
130:0    0 {
131:0    0 {
132:0    0 {
133:0    0 {
134:0    0 {
135:0    0 {
136:0    0 {
137:0    0 {
138:0    0 {
139:0    0 {
140:0    0 {
141:0    0 {
142:0    0 {
143:0    0 {
144:0    0 {
145:0    0 {
146:0    0 {
147:0    0 {
148:0    0 {
149:0    0 {
150:0    0 {
151:0    0 {
152:0    0 {
153:0    0 {
154:0    0 {
155:0    0 {
156:0    0 {
157:0    0 {
158:0    0 {
159:0    0 {
160:0    0 {
161:0    0 {
162:0    0 {
163:0    0 {
164:0    0 {
165:0    0 {
166:0    0 {
167:0    0 {
168:0    0 {
169:0    0 {
170:0    0 {
171:0    0 {
172:0    0 {
173:0    0 {
174:0    0 {
175:0    0 {
176:0    0 {
177:0    0 {
178:0    0 {
179:0    0 {
180:0    0 {
181:0    0 {
182:0    0 {
183:0    0 {
184:0    0 {
185:0    0 {
186:0    0 {
187:0    0 {
188:0    0 {
189:0    0 {
190:0    0 {
191:0    0 {
192:0    0 {
193:0    0 {
194:0    0 {
195:0    0 {
196:0    0 {
197:0    0 {
198:0    0 {
199:0    0 {
200:0    0 {
201:0    0 {
202:0    0 {
203:0    0 {
204:0    0 {
205:0    0 {
206:0    0 {
207:0    0 {
208:0    0 {
209:0    0 {
210:0    0 {
211:0    0 {
212:0    0 {
213:0    0 {
214:0    0 {
215:0    0 {
216:0    0 {
217:0    0 {
218:0    0 {
219:0    0 {
220:0    0 {
221:0    0 {
222:0    0 {
223:0    0 {
224:0    0 {
225:0    0 {
226:0    0 {
227:0    0 {
228:0    0 {
229:0    0 {
230:0    0 {
231:0    0 {
232:0    0 {
233:0    0 {
234:0    0 {
235:0    0 {
236:0    0 {
237:0    0 {
238:0    0 {
239:0    0 {
240:0    0 {
241:0    0 {
242:0    0 {
243:0    0 {
244:0    0 {
245:0    0 {
246:0    0 {
247:0    0 {
248:0    0 {
249:0    0 {
250:0    0 {
251:0    0 {
252:0    0 {
253:0    0 {
254:0    0 {
255:0    0 {
256:0    0 {
257:0    0 {
258:0    0 {
259:0    0 {
260:0    0 {
261:0    0 {
262:0    0 {
263:0    0 {
264:0    0 {
265:0    0 {
266:0    0 {
267:0    0 {
268:0    0 {
269:0    0 {
270:0    0 {
271:0    0 {
272:0    0 {
273:0    0 {
274:0    0 {
275:0    0 {
276:0    0 {
277:0    0 {
278:0    0 {
279:0    0 {
280:0    0 {
281:0    0 {
282:0    0 {
283:0    0 {
284:0    0 {
285:0    0 {
286:0    0 {
287:0    0 {
288:0    0 {
289:0    0 {
290:0    0 {
291:0    0 {
292:0    0 {
293:0    0 {
294:0    0 {
295:0    0 {
296:0    0 {
297:0    0 {
298:0    0 {
299:0    0 {
300:0    0 {
301:0    0 {
302:0    0 {
303:0    0 {
304:0    0 {
305:0    0 {
306:0    0 {
307:0    0 {
308:0    0 {
309:0    0 {
310:0    0 {
311:0    0 {
312:0    0 {
313:0    0 {
314:0    0 {
315:0    0 {
316:0    0 {
317:0    0 {
318:0    0 {
319:0    0 {
320:0    0 {
321:0    0 {
322:0    0 {
323:0    0 {
324:0    0 {
325:0    0 {
326:0    0 {
327:0    0 {
328:0    0 {
329:0    0 {
330:0    0 {
331:0    0 {
332:0    0 {
333:0    0 {
334:0    0 {
335:0    0 {
336:0    0 {
337:0    0 {
338:0    0 {
339:0    0 {
340:0    0 {
341:0    0 {
342:0    0 {
343:0    0 {
344:0    0 {
345:0    0 {
346:0    0 {
347:0    0 {
348:0    0 {
349:0    0 {
350:0    0 {
351:0    0 {
352:0    0 {
353:0    0 {
354:0    0 {
355:0    0 {
356:0    0 {
357:0    0 {
358:0    0 {
359:0    0 {
360:0    0 {
361:0    0 {
362:0    0 {
363:0    0 {
364:0    0 {
365:0    0 {
366:0    0 {
367:0    0 {
368:0    0 {
369:0    0 {
370:0    0 {
371:0    0 {
372:0    0 {
373:0    0 {
374:0    0 {
375:0    0 {
376:0    0 {
377:0    0 {
378:0    0 {
379:0    0 {
380:0    0 {
381:0    0 {
382:0    0 {
383:0    0 {
384:0    0 {
385:0    0 {
386:0    0 {
387:0    0 {
388:0    0 {
389:0    0 {
390:0    0 {
391:0    0 {
392:0    0 {
393:0    0 {
394:0    0 {
395:0    0 {
396:0    0 {
397:0    0 {
398:0    0 {
399:0    0 {
400:0    0 {
401:0    0 {
402:0    0 {
403:0    0 {
404:0    0 {
405:0    0 {
406:0    0 {
407:0    0 {
408:0    0 {
409:0    0 {
410:0    0 {
411:0    0 {
412:0    0 {
413:0    0 {
414:0    0 {
415:0    0 {
416:0    0 {
417:0    0 {
418:0    0 {
419:0    0 {
420:0    0 {
421:0    0 {
422:0    0 {
423:0    0 {
424:0    0 {
425:0    0 {
426:0    0 {
427:0    0 {
428:0    0 {
429:0    0 {
430:0    0 {
431:0    0 {
432:0    0 {
433:0    0 {
434:0    0 {
435:0    0 {
436:0    0 {
437:0    0 {
438:0    0 {
439:0    0 {
440:0    0 {
441:0    0 {
442:0    0 {
443:0    0 {
444:0    0 {
445:0    0 {
446:0    0 {
447:0    0 {
448:0    0 {
449:0    0 {
450:0    0 {
451:0    0 {
452:0    0 {
453:0    0 {
454:0    0 {
455:0    0 {
456:0    0 {
457:0    0 {
458:0    0 {
459:0    0 {
460:0    0 {
461:0    0 {
462:0    0 {
463:0    0 {
464:0    0 {
465:0    0 {
466:0    0 {
467:0    0 {
468:0    0 {
469:0    0 {
470:0    0 {
471:0    0 {
472:0    0 {
473:0    0 {
474:0    0 {
475:0    0 {
476:0    0 {
477:0    0 {
478:0    0 {
479:0    0 {
480:0    0 {
481:0    0 {
482:0    0 {
483:0    0 {
484:0    0 {
485:0    0 {
486:0    0 {
487:0    0 {
488:0    0 {
489:0    0 {
490:0    0 {
491:0    0 {
492:0    0 {
493:0    0 {
494:0    0 {
495:0    0 {
496:0    0 {
497:0    0 {
498:0    0 {
499:0    0 {
500:0    0 {
501:0    0 {
502:0    0 {
503:0    0 {
504:0    0 {
505:0    0 {
506:0    0 {
507:0    0 {
508:0    0 {
509:0    0 {
510:0    0 {
511:0    0 {
512:0    0 {
513:0    0 {
514:0    0 {
515:0    0 {
516:0    0 {
517:0    0 {
518:0    0 {
519:0    0 {
520:0    0 {
521:0    0 {
522:0    0 {
523:0    0 {
524:0    0 {
525:0    0 {
526:0    0 {
527:0    0 {
528:0    0 {
529:0    0 {
530:0    0 {
531:0    0 {
532:0    0 {
533:0    0 {
534:0    0 {
535:0    0 {
536:0    0 {
537:0    0 {
538:0    0 {
539:0    0 {
540:0    0 {
541:0    0 {
542:0    0 {
543:0    0 {
544:0    0 {
545:0    0 {
546:0    0 {
547:0    0 {
548:0    0 {
549:0    0 {
550:0    0 {
551:0    0 {
552:0    0 {
553:0    0 {
554:0    0 {
555:0    0 {
556:0    0 {
557:0    0 {
558:0    0 {
559:0    0 {
560:0    0 {
561:0    0 {
562:0    0 {
563:0    0 {
564:0    0 {
565:0    0 {
566:0    0 {
567:0    0 {
568:0    0 {
569:0    0 {
570:0    0 {
571:0    0 {
572:0    0 {
573:0    0 {
574:0    0 {
575:0    0 {
576:0    0 {
577:0    0 {
578:0    0 {
579:0    0 {
580:0    0 {
581:0    0 {
582:0    0 {
583:0    0 {
584:0    0 {
585:0    0 {
586:0    0 {
587:0    0 {
588:0    0 {
589:0    0 {
590:0    0 {
591:0    0 {
592:0    0 {
593:0    0 {
594:0    0 {
595:0    0 {
596:0    0 {
597:0    0 {
598:0    0 {
599:0    0 {
600:0    0 {
601:0    0 {
602:0    0 {
603:0    0 {
604:0    0 {
605:0    0 {
606:0    0 {
607:0    0 {
608:0    0 {
609:0    0 {
610:0    0 {
611:0    0 {
612:0    0 {
613:0    0 {
614:0    0 {
615:0    0 {
616:0    0 {
617:0    0 {
618:0    0 {
619:0    0 {
620:0    0 {
621:0    0 {
622:0    0 {
623:0    0 {
624:0    0 {
625:0    0 {
626:0    0 {
627:0    0 {
628:0    0 {
629:0    0 {
630:0    0 {
631:0    0 {
632:0    0 {
633:0    0 {
634:0    0 {
635:0    0 {
636:0    0 {
637:0    0 {
638:0    0 {
639:0    0 {
640:0    0 {
641:0    0 {
642:0    0 {
643:0    0 {
644:0    0 {
645:0    0 {
646:0    0 {
647:0    0 {
648:0    0 {
649:0    0 {
650:0    0 {
651:0    0 {
652:0    0 {
653:0    0 {
654:0    0 {
655:0    0 {
656:0    0 {
657:0    0 {
658:0    0 {
659:0    0 {
660:0    0 {
661:0    0 {
662:0    0 {
663:0    0 {
664:0    0 {
665:0    0 {
666:0    0 {
667:0    0 {
668:0    0 {
669:0    0 {
670:0    0 {
671:0    0 {
672:0    0 {
673:0    0 {
674:0    0 {
675:0    0 {
676:0    0 {
677:0    0 {
678:0    0 {
679:0    0 {
680:0    0 {
681:0    0 {
682:0    0 {
683:0    0 {
684:0    0 {
685:0    0 {
686:0    0 {
687:0    0 {
688:0    0 {
689:0    0 {
690:0    0 {
691:0    0 {
692:0    0 {
693:0    0 {
694:0    0 {
695:0    0 {
696:0    0 {
697:0    0 {
698:0    0 {
699:0    0 {
700:0    0 {
701:0    0 {
702:0    0 {
703:0    0 {
704:0    0 {
705:0    0 {
706:0    0 {
707:0    0 {
708:0    0 {
709:0    0 {
710:0    0 {
711:0    0 {
712:0    0 {
713:0    0 {
714:0    0 {
715:0    0 {
716:0    0 {
717:0    0 {
718:0    0 {
719:0    0 {
720:0    0 {
721:0    0 {
722:0    0 {
723:0    0 {
724:0    0 {
725:0    0 {
726:0    0 {
727:0    0 {
728:0    0 {
729:0    0 {
730:0    0 {
731:0    0 {
732:0    0 {
733:0    0 {
734:0    0 {
735:0    0 {
736:0    0 {
737:0    0 {
738:0    0 {
739:0    0 {
740:0    0 {
741:0    0 {
742:0    0 {
743:0    0 {
744:0    0 {
745:0    0 {
746:0    0 {
747:0    0 {
748:0    0 {
749:0    0 {
750:0    0 {
751:0    0 {
752:0    0 {
753:0    0 {
754:0    0 {
755:0    0 {
756:0    0 {
757:0    0 {
758:0    0 {
759:0    0 {
760:0    0 {
761:0    0 {
762:0    0 {
763:0    0 {
764:0    0 {
765:0    0 {
766:0    0 {
767:0    0 {
768:0    0 {
769:0    0 {
770:0    0 {
771:0    0 {
772:0    0 {
773:0    0 {
774:0    0 {
775:0    0 {
776:0    0 {
777:0    0 {
778:0    0 {
779:0    0 {
780:0    0 {
781:0    0 {
782:0    0 {
783:0    0 {
784:0    0 {
785:0    0 {
786:0    0 {
787:0    0 {
788:0    0 {
789:0    0 {
790:0    0 {
791:0    0 {
792:0    0 {
793:0    0 {
794:0    0 {
795:0    0 {
796:0    0 {
797:0    0 {
798:0    0 {
799:0    0 {
800:0    0 {
801:0    0 {
802:0    0 {
803:0    0 {
804:0    0 {
805:0    0 {
806:0    0 {
807:0    0 {
808:0    0 {
809:0    0 {
810:0    0 {
811:0    0 {
812:0    0 {
813:0    0 {
814:0    0 {
815:0    0 {
816:0    0 {
817:0    0 {
818:0    0 {
819:0    0 {
820:0    0 {
821:0    0 {
822:0    0 {
823:0    0 {
824:0    0 {
825:0    0 {
826:0    0 {
827:0    0 {
828:0    0 {
829:0    0 {
830:0    0 {
831:0    0 {
832:0    0 {
833:0    0 {
834:0    0 {
835:0    0 {
836:0    0 {
837:0    0 {
838:0    0 {
839:0    0 {
840:0    0 {
841:0    0 {
842:0    0 {
843:0    0 {
844:0    0 {
845:0    0 {
846:0    0 {
847:0    0 {
848:0    0 {
849:0    0 {
850:0    0 {
851:0    0 {
852:0    0 {
853:0    0 {
854:0    0 {
855:0    0 {
856:0    0 {
857:0    0 {
858:0    0 {
859:0    0 {
860:0    0 {
861:0    0 {
862:0    0 {
863:0    0 {
864:0    0 {
865:0    0 {
866:0    0 {
867:0    0 {
868:0    0 {
869:0    0 {
870:0    0 {
871:0    0 {
872:0    0 {
873:0    0 {
874:0    0 {
875:0    0 {
876:0    0 {
877:0    0 {
878:0    0 {
879:0    0 {
880:0    0 {
881:0    0 {
882:0    0 {
883:0    0 {
884:0    0 {
885:0    0 {
886:0    0 {
887:0    0 {
888:0    0 {
889:0    0 {
890:0    0 {
891:0    0 {
892:0    0 {
893:0    0 {
894:0    0 {
895:0    0 {
896:0    0 {
897:0    0 {
898:0    0 {
899:0    0 {
900:0    0 {
901:0    0 {
902:0    0 {
903:0    0 {
904:0    0 {
905:0    0 {
906:0    0 {
907:0    0 {
908:0    0 {
909:0    0 {
910:0    0 {
911:0    0 {
912:0    0 {
913:0    0 {
914:0    0 {
915:0    0 {
916:0    0 {
917:0    0 {
918:0    0 {
919:0    0 {
920:0    0 {
921:0    0 {
922:0    0 {
923:0    0 {
924:0    0 {
925:0    0 {
926:0    0 {
927:0    0 {
928:0    0 {
929:0    0 {
930:0    0 {
931:0    0 {
932:0    0 {
933:0    0 {
934:0    0 {
935:0    0 {
936:0    0 {
937:0    0 {
938:0    0 {
939:0    0 {
940:0    0 {
941:0    0 {
942:0    0 {
943:0    0 {
944:0    0 {
945:0    0 {
946:0    0 {
947:0    0 {
948:0    0 {
949:0    0 {
950:0    0 {
951:0    0 {
952:0    0 {
953:0    0 {
954:0    0 {
955:0    0 {
956:0    0 {
957:0    0 {
958:0    0 {
959:0    0 {
960:0    0 {
961:0    0 {
962:0    0 {
963:0    0 {
964:0    0 {
965:0    0 {
966:0    0 {
967:0    0 {
968:0    0 {
969:0    0 {
970:0    0 {
971:0    0 {
972:0    0 {
973:0    0 {
974:0    0 {
975:0    0 {
976:0    0 {
977:0    0 {
978:0    0 {
979:0    0 {
980:0    0 {
981:0    0 {
982:0    0 {
983:0    0 {
984:0    0 {
985:0    0 {
986:0    0 {
987:0    0 {
988:0    0 {
989:0    0 {
990:0    0 {
991:0    0 {
992:0    0 {
993:0    0 {
994:0    0 {
995:0    0 {
996:0    0 {
997:0    0 {
998:0    0 {
999:0    0 {
1000:0   0 {

```

```

6014:0    2      anyvar rlist : greal_list;
6015:0    2      var ierr : integer;
6016:0    2
6017:0    1 procedure inq_color_table ( index : integer;
6018:0    2      var parm1 : real;
6019:0    2      var parm2 : real;
6020:0    2      var parm3 : real);
6021:0    2
6022:0    1 procedure inq_pgn_table ( index : integer;
6023:0    2      var pdensity : real;
6024:0    2      var porient : real;
6025:0    2      var pedge : integer);
6026:0    2
6027:0    1 implement
6028:0    2
6029:0    1 import dgl_vars,
6030:0    1      gle_has_out,
6031:0    1      dgl_gen;
6032:0    2
6033:0    1 procedure inq_color_table ( index : integer;
6034:0    2      var parm1 : real;
6035:0    2      var parm2 : real;
6036:0    2      var parm3 : real);
6037:0    2
6038:0    2 { Purpose: To return information about a color table location }
6039:0    2
6040:0    2
6041:0    2 begin
6042:0    2 ck_system_init;
6043:0    2 ck_display_init;
6044:0    2 with gcb^ do
6045:0    2 begin
6046:0    2 if color_table_size = 0 then error(err_no_ctable);
6047:0    2 if (index < 0) or (index > color_table_size) then error(err_bad_parms);
6048:0    2 with color_table_ptr^[index] do
6049:0    2 if dgl_current_color_model = 1 { rgb } then
6050:0    2 begin
6051:0    2 parm1 := red;
6052:0    2 parm2 := green;
6053:0    2 parm3 := blue;
6054:0    2 end
6055:0    2 else
6056:0    2 convert_rgb_to_hsl(red,green,blue,parm1,parm2,parm3);
6057:0    2 end;
6058:0    2 end;
6059:0    2
6060:0    1 procedure inq_pgn_table ( index : integer;
6061:0    2      var pdensity : real;
6062:0    2      var porient : real;
6063:0    2      var pedge : integer);
6064:0    2
6065:0    2 { Purpose: To return information about a polygon table location }
6066:0    2
6067:0    2
6068:0    2 begin
6069:0    2 ck_system_init;
6070:0    2 ck_display_init;
6071:0    2 with gcb^ do
6072:0    2 begin
6073:0    2 if (index < 1) or (index > number_polygon_styles) then error(err_bad_parms);

```



```

6074:C      3      with poly_table_ptr^ [ index ] do
6075:C      4      begin
6076:C      4          pdensity := density;
6077:C      4          porient := orient;
6078:C      4          if edge then pedge := 1
6079:C      5          else pedge := 0;
6080:C      4      end;
6081:C      2      end;
6082:C      2      end;
6083:S
6084:D      1      procedure inq_ws (          opcode : integer;
6085:D      2          isize : integer;
6086:D      2          rsize : integer;
6087:D      2          anyvar slist : gchar_list;
6088:D      2          anyvar rlist : gnt_list;
6089:D      2          anyvar rlist : greal_list;
6090:D      2          var ierr : integer);
6091:D      2
6092:S
6093:D      2      ( Purpose: To return information some part of the DGL graphics system. )
6094:S
6095:D      2      label 1;
6096:S
6097:D      2      const numopcodes = 40;
6098:S
6099:D      2      type
6100:D      2          opcode_type_def = (inq_250,   inq_251,   inq_252,   inq_253,   inq_254,
6101:D      2          inq_255,   inq_256,   inq_257,   inq_258,   inq_259,
6102:D      2          inq_450,   inq_451,
6103:D      2          inq_1050,   inq_1051,   inq_1052,   inq_1053,   inq_1054,
6104:D      2          inq_1055,   inq_1056,   inq_1057,
6105:D      2          inq_1060,   inq_1061,   inq_1062,   inq_1063,   inq_1064,
6106:D      2          inq_1065,   inq_1066,   inq_1067,   inq_1068,   inq_1069,
6107:D      2          inq_1070,   inq_1071,   inq_1072,   inq_1073,   inq_1074,
6108:D      2          inq_1075,   inq_1076,
6109:D      2          inq_11050,   inq_11052,
6110:D      2          inq_12050,
6111:D      2          inq_13052 );
6112:S
6113:D      2      opcode_def = 0..32767;
6114:D      2      opcode_list_entry_def = packed record
6115:D      2          opcode : opcode_def;
6116:D      2          opcode_type : opcode_type_def;
6117:D      2      end;
6118:D      2      opcode_list_def = packed array [1..numopcodes] of opcode_list_entry_def;
6119:S
6120:D      2      const
6121:D      2          opcode_list = opcode_list_def[
6122:D      2          opcode_list_entry_def(opcode : 250,   opcode_type : inq_250 ),
6123:D      2          opcode_list_entry_def(opcode : 251,   opcode_type : inq_251 ),
6124:D      2          opcode_list_entry_def(opcode : 252,   opcode_type : inq_252 ),
6125:D      2          opcode_list_entry_def(opcode : 253,   opcode_type : inq_253 ),
6126:D      2          opcode_list_entry_def(opcode : 254,   opcode_type : inq_254 ),
6127:D      2          opcode_list_entry_def(opcode : 255,   opcode_type : inq_255 ),
6128:D      2          opcode_list_entry_def(opcode : 256,   opcode_type : inq_256 ),
6129:D      2          opcode_list_entry_def(opcode : 257,   opcode_type : inq_257 ),
6130:D      2          opcode_list_entry_def(opcode : 258,   opcode_type : inq_258 ),
6131:D      2          opcode_list_entry_def(opcode : 259,   opcode_type : inq_259 ),
6132:D      2          opcode_list_entry_def(opcode : 450,   opcode_type : inq_450 ),
6133:D      2          opcode_list_entry_def(opcode : 451,   opcode_type : inq_451 )];

```

```

6134:D      2          opcode_list_entry_def(opcode : 1050, opcode_type : inq_1050 ),
6135:D      2          opcode_list_entry_def(opcode : 1051, opcode_type : inq_1051 ),
6136:D      2          opcode_list_entry_def(opcode : 1052, opcode_type : inq_1052 ),
6137:D      2          opcode_list_entry_def(opcode : 1053, opcode_type : inq_1053 ),
6138:D      2          opcode_list_entry_def(opcode : 1054, opcode_type : inq_1054 ),
6139:D      2          opcode_list_entry_def(opcode : 1055, opcode_type : inq_1055 ),
6140:D      2          opcode_list_entry_def(opcode : 1056, opcode_type : inq_1056 ),
6141:D      2          opcode_list_entry_def(opcode : 1057, opcode_type : inq_1057 ),
6142:D      2          opcode_list_entry_def(opcode : 1058, opcode_type : inq_1058 ),
6143:D      2          opcode_list_entry_def(opcode : 1059, opcode_type : inq_1059 ),
6144:D      2          opcode_list_entry_def(opcode : 1060, opcode_type : inq_1060 ),
6145:D      2          opcode_list_entry_def(opcode : 1061, opcode_type : inq_1061 ),
6146:D      2          opcode_list_entry_def(opcode : 1062, opcode_type : inq_1062 ),
6147:D      2          opcode_list_entry_def(opcode : 1063, opcode_type : inq_1063 ),
6148:D      2          opcode_list_entry_def(opcode : 1064, opcode_type : inq_1064 ),
6149:D      2          opcode_list_entry_def(opcode : 1065, opcode_type : inq_1065 ),
6150:D      2          opcode_list_entry_def(opcode : 1066, opcode_type : inq_1066 ),
6151:D      2          opcode_list_entry_def(opcode : 1067, opcode_type : inq_1067 ),
6152:D      2          opcode_list_entry_def(opcode : 1068, opcode_type : inq_1068 ),
6153:D      2          opcode_list_entry_def(opcode : 1069, opcode_type : inq_1069 ),
6154:D      2          opcode_list_entry_def(opcode : 1070, opcode_type : inq_1070 ),
6155:D      2          opcode_list_entry_def(opcode : 1071, opcode_type : inq_1071 ),
6156:D      2          opcode_list_entry_def(opcode : 1072, opcode_type : inq_1072 ),
6157:D      2          opcode_list_entry_def(opcode : 1073, opcode_type : inq_1073 ),
6158:D      2          opcode_list_entry_def(opcode : 1074, opcode_type : inq_1074 ),
6159:D      2          opcode_list_entry_def(opcode : 1075, opcode_type : inq_1075 ),
6160:D      2          opcode_list_entry_def(opcode : 1076, opcode_type : inq_1076 ),
6161:D      2          opcode_list_entry_def(opcode : 11050, opcode_type : inq_11050 ),
6162:D      2          opcode_list_entry_def(opcode : 11052, opcode_type : inq_11052 ),
6163:D      2          opcode_list_entry_def(opcode : 12050, opcode_type : inq_12050 ),
6164:D      2          opcode_list_entry_def(opcode : 13052, opcode_type : inq_13052 );
6165:S
6166:D      2      var
6167:D      2          index : gshortint;
6168:D      2          workstring : string[10];
6169:D      2          strcnt : integer;
6170:S
6171:D      2      begin
6172:D      2          ck_system_init;
6173:S
6174:D      2          ierr := opcode_ck ( opcode, isize, rsize ); { ck for good parms }
6175:D      2
6176:D      2          { Find opcode in list }
6177:D      2          for index := 1 to numopcodes do
6178:D      2              if opcode_list[index].opcode = opcode then goto 1;
6179:D      2          ierr := 1;
6180:S
6181:D      2      1:
6182:D      2          if ierr = 0 then
6183:D      2              with gcb^, gle_gcb^ do
6184:D      2                  case opcode_list[index].opcode_type of
6185:D      2                      inq_250 : ( return cell size (250) )
6186:D      2                      begin
6187:D      2                          rlist[1] := gcb^.dgl_char_width;
6188:D      2                          rlist[2] := gcb^.dgl_char_height;
6189:D      2                      end;
6190:D      2                      inq_251 : ( return marker cell size (251) )
6191:D      2                      begin
6192:D      2                          rlist[1] := marker_size_x * xdtow_scale;
6193:D      2                          rlist[2] := marker_size_y * ydtow_scale;

```



```

6194:C      5      inq_252 : { return display resolution (252) }
6195:C      5      if disp_init then
6196:C      6      begin
6197:C      6      with gcb^, raster_device_rec_ptr(dev_dep_stuff)^ do
6198:C      7      begin
6200:C      7      rlist[1] := display_res_x;
6201:C      7      rlist[2] := display_res_y;
6202:C      7      end;
6203:C      6      end;
6204:C      6      else
6205:C      6      begin
6206:C      6      rlist[1] := 0;
6207:C      6      rlist[2] := 0;
6208:C      6      end;
6209:C      5      inq_253 : { return max display dimensions (253) }
6210:C      5      if disp_init then
6211:C      6      with max_disp_lim do
6212:C      7      begin
6213:C      7      with gcb^, raster_device_rec_ptr(dev_dep_stuff)^ do
6214:C      8      begin
6215:C      8      rlist[1] := (xmax - xmin) / display_res_x;
6216:C      8      rlist[2] := (ymax - ymin) / display_res_y;
6217:C      8      end;
6218:C      7      end;
6219:C      7      else
6220:C      6      begin
6221:C      6      rlist[1] := 0;
6222:C      6      rlist[2] := 0;
6223:C      6      end;
6224:C      5      inq_254 : { return aspect ratios (254) }
6225:C      5      begin
6226:C      5      rlist[1] := aspect_ratio;
6227:C      5      rlist[2] := log_aspect;
6228:C      5      end;
6229:C      5      inq_255 : { return locator resolution (255) }
6230:C      5      if loc_init then
6231:C      6      with gle_gcb1^ do
6232:C      7      begin
6233:C      7      rlist[1] := input_res_x;
6234:C      7      rlist[2] := input_res_y;
6235:C      7      end;
6236:C      6      else
6237:C      6      begin
6238:C      6      rlist[1] := 0;
6239:C      6      rlist[2] := 0;
6240:C      6      end;
6241:C      5      inq_256 : { max locator dimensions (256) }
6242:C      5      if loc_init then
6243:C      6      with gle_gcb1^, max_loc_lim do
6244:C      7      begin
6245:C      7      rlist[1] := (xmax - xmin) / input_res_x;
6246:C      7      rlist[2] := (ymax - ymin) / input_res_y;
6247:C      7      end;
6248:C      6      else
6249:C      6      begin

```

```

6254:C      6      rlist[1] := 0;
6255:C      6      rlist[2] := 0;
6256:C      6      end;
6257:C      5      inq_257 : { locator echo pos (257) }
6258:C      5      begin
6259:C      5      rlist[1] := w_loc_echo_x;
6260:C      5      rlist[2] := w_loc_echo_y;
6261:C      5      end;
6262:C      5      inq_258 : { current virtual limits (258) }
6263:C      5      with cur_vir_lim do
6264:C      6      begin
6265:C      6      rlist[1] := xlim;
6266:C      6      rlist[2] := ylim;
6267:C      6      end;
6268:C      5      inq_259 : { return cp (259) }
6269:C      5      begin
6270:C      6      if int_cp then
6271:C      7      begin
6272:C      7      rlist[1] := world_int_cpx;
6273:C      7      rlist[2] := world_int_cpy;
6274:C      7      end;
6275:C      6      else
6276:C      6      begin
6277:C      6      rlist[1] := world_real_cpx;
6278:C      6      rlist[2] := world_real_cpy;
6279:C      6      end;
6280:C      5      end;
6281:C      5      inq_450 : { return window limits (450) }
6282:C      5      with window_lim do
6283:C      6      begin
6284:C      6      rlist[1] := xmin;
6285:C      6      rlist[2] := xmax;
6286:C      6      rlist[3] := ymin;
6287:C      6      rlist[4] := ymax;
6288:C      6      end;
6289:C      5      inq_451 : { return viewport limits (451) }
6290:C      5      with viewport_lim do
6291:C      6      begin
6292:C      6      rlist[1] := xmin;
6293:C      6      rlist[2] := xmax;
6294:C      6      rlist[3] := ymin;
6295:C      6      rlist[4] := ymax;
6296:C      6      end;
6297:C      5      inq_1050 : { Clipping supported at physical limits (1050) }
6298:C      5      begin
6299:C      6      if clipping_support then ilist[1] := 1;
6300:C      6      else ilist[1] := 0;
6301:C      6      end;
6302:C      5      inq_1051 : { return display justification info (1051) }
6303:C      5      begin
6304:C      6      if disp_just = centered then ilist[1] := 0;
6305:C      6      else ilist[1] := 1;
6306:C      6      end;

```



```

6314:S      5      inq_1052: ( return info about drawing in the background color (1052) )
6315:C      5      with gcb do
6316:C      6      begin
6317:C      6          if disp_init then ilist[1] := background
6318:C      7          else ilist[1] := 0;
6319:C      6      end;
6320:C      5
6321:S      5      inq_1053: ( return color palette info (1053) )
6322:C      5      begin
6323:C      5          if disp_init then ilist[1] := palette
6324:C      6          else ilist[1] := 0;
6325:C      5      end;
6326:C      5
6327:S      5      inq_1054: ( return color gamut info (1054) )
6328:C      5      begin
6329:C      5          if disp_init then ilist[1] := gamut
6330:C      6          else ilist[1] := 0;
6331:C      5      end;
6332:C      5
6333:S      5      inq_1056: ( return number linestyle (1056) )
6334:C      5      begin
6335:C      5          if disp_init then ilist[1] := number_dgl_linestyles
6336:C      6          else ilist[1] := 0;
6337:C      5      end;
6338:C      5
6339:S      5      inq_1057: ( return number linewidths (1057) )
6340:C      5      begin
6341:C      5          if disp_init then ilist[1] := linewidths
6342:C      6          else ilist[1] := 0;
6343:C      5      end;
6344:C      5
6345:S      5      inq_1059: ( return number markers (1059) )
6346:C      5      begin
6347:C      5          if disp_init then ilist[1] := number_markers
6348:C      6          else ilist[1] := 0;
6349:C      5      end;
6350:C      5
6351:S      5      inq_1060: ( return current color (1060) )
6352:C      5      begin
6353:C      5          ilist[1] := dgl_current_color;
6354:C      5      end;
6355:C      5
6356:S      5      inq_1062: ( return current linestyle (1062) )
6357:C      5      begin
6358:C      5          ilist[1] := dgl_current_linestyle;
6359:C      5      end;
6360:C      5
6361:S      5      inq_1063: ( return current linewidth (1063) )
6362:C      5      begin
6363:C      5          ilist[1] := dgl_current_linewidth;
6364:C      5      end;
6365:C      5
6366:S      5      inq_1064: ( return current timing mode )
6367:C      5      begin
6368:C      5          ilist[1] := dgl_current_timing_mode;
6369:C      5      end;
6370:C      5
6371:S      5      inq_1065: ( return number polygon styles supported ( 1065) )
6372:C      5      begin
6373:C      5

```

```

6374:C      5      if disp_init then ilist[1] := number_polygon_styles
6375:C      6      else ilist[1] := 0;
6376:C      5      end;
6377:S      5      inq_1066: ( return current polygon color (1066) )
6378:C      5      begin
6379:C      5          ilist[1] := dgl_current_polygon_color;
6380:C      5      end;
6381:C      5
6382:S      5      inq_1067: ( return current polygon style (1067) )
6383:C      5      begin
6384:C      5          ilist[1] := dgl_current_polygon_style;
6385:C      5      end;
6386:C      5
6387:S      5      inq_1068: ( return maximum polygon vertices supported (1068) )
6388:C      5      begin
6389:C      5          if disp_init then ilist[1] := maximum_polygon_vertices
6390:C      6          else ilist[1] := 0;
6391:C      5      end;
6392:C      5
6393:S      5      inq_1069: ( retroactive polygon support (1069) )
6394:C      5      begin
6395:C      5          if disp_init and retroactive_polygon_support then ilist[1] := 1
6396:C      6          else ilist[1] := 0;
6397:C      5      end;
6398:C      5
6399:S      5      inq_1070: ( device dependent polygons ( 1070 ) )
6400:C      5      begin
6401:C      5          if disp_init and (polygon_support = 1) then ilist[1] := 1
6402:C      6          else ilist[1] := 0;
6403:C      5      end;
6404:C      5
6405:S      5      inq_1071: ( retroactive color support ( 1071 ) )
6406:C      5      begin
6407:C      5          if disp_init and retroactive_color_support then ilist[1] := 1
6408:C      6          else ilist[1] := 0;
6409:C      5      end;
6410:C      5
6411:S      5      inq_1072: ( redef of background ( 1072 ) )
6412:C      5      begin
6413:C      5          if disp_init then ilist[1] := redef_background
6414:C      6          else ilist[1] := 0;
6415:C      5      end;
6416:C      5
6417:S      5      inq_1073: ( redef of color capability table ( 1073 ) )
6418:C      5      begin
6419:C      5          if disp_init and (color_table_size > 0) then ilist[1] := 1
6420:C      6          else ilist[1] := 0;
6421:C      5      end;
6422:C      5
6423:S      5      inq_1074: ( return current color model ( 1074 ) )
6424:C      5      begin
6425:C      5          ilist[1] := dgl_current_color_model;
6426:C      5      end;
6427:C      5
6428:S      5      inq_1075: ( return color capability table size ( 1075 ) )
6429:C      5      ilist[1] := color_table_size;
6430:C      5
6431:S      5      inq_1076: ( return current polygon linestyle ( 1076 ) )
6432:C      5      ilist[1] := dgl_current_polygon_linestyle;
6433:C      5

```



```
6434:0      5      inq_11050: ( return display device association (11050) )
6435:0      5      begin
6436:0      5      if not disp_init then
6437:0      5      begin
6438:0      6      if ssize >= 1 then
6439:0      6      begin
6440:0      7      slist[1] := '0';
6441:0      7      ilist[1] := 1;
6442:0      7      end
6443:0      6      else ierr := 4;
6444:0      6      end
6445:0      6      else
6446:0      6      if disp_file_name <> '' then
6447:0      6      begin
6448:0      7      for index := 1 to strlen(disp_file_name) do
6449:0      7      slist[index] := disp_file_name[index];
6450:0      7      ilist[index] := strlen(disp_file_name);
6451:0      7      end
6452:0      6      else
6453:0      6      begin
6454:0      7      setstrlen(workstring,0);
6455:0      7      strwrite(workstring,1,strtnt,disp_dev_adr:0);
6456:0      7      strtnt := strtnt - 1;
6457:0      7      if ssize >= strtnt then
6458:0      8      begin
6459:0      9      for index := 1 to strtnt do
6460:0      9      slist[index] := workstring[index];
6461:0      9      ilist[1] := strtnt;
6462:0      8      end
6463:0      6      else
6464:0      6      ierr := 4;
6465:0      6      end;
6466:0      5      end;
6467:0      5
6468:0      5
6469:0      5      inq_11052: ( return locator device association (11052) )
6470:0      5      begin
6471:0      5      if not loc_init then
6472:0      5      begin
6473:0      6      if ssize >= 1 then
6474:0      6      begin
6475:0      7      slist[1] := '0';
6476:0      7      ilist[1] := 1;
6477:0      7      end
6478:0      6      else ierr := 4;
6479:0      6      end
6480:0      6      else
6481:0      6      begin
6482:0      7      setstrlen(workstring,0);
6483:0      7      strwrite(workstring,1,strtnt,loc_dev_adr:0);
6484:0      7      strtnt := strtnt - 1;
6485:0      7      if ssize >= strtnt then
6486:0      8      begin
6487:0      9      for index := 1 to strtnt do
6488:0      9      slist[index] := workstring[index];
6489:0      9      ilist[1] := strtnt;
6490:0      8      end
6491:0      6      else
6492:0      6      ierr := 4;
6493:0      5      end;
6494:0      5
```

```
6494:0      5      end;
6495:0      5
6496:0      5      inq_12050: ( return display info (12050) )
6497:0      5      begin
6498:0      5      if not disp_init then
6499:0      5      begin
6500:0      6      if ssize >= 6 then
6501:0      6      begin
6502:0      7      for index := 1 to 6 do
6503:0      7      slist[index] := ' ';
6504:0      7      ilist[1] := 6;
6505:0      7      ilist[2] := 0; ( disabled )
6506:0      7      end
6507:0      6      else ierr := 4;
6508:0      6      end
6509:0      6      else
6510:0      6      begin
6511:0      7      if ssize >= display_name_char_count then
6512:0      7      begin
6513:0      8      for index := 1 to display_name_char_count do
6514:0      8      slist[index] := display_name[index];
6515:0      8      ilist[1] := display_name_char_count;
6516:0      8      ilist[2] := 1; ( enabled )
6517:0      7      end
6518:0      6      else
6519:0      6      ierr := 4;
6520:0      6      end;
6521:0      5      end;
6522:0      5
6523:0      5      inq_13052: ( return locator info (13052) )
6524:0      5      begin
6525:0      5      if not loc_init then
6526:0      5      begin
6527:0      6      if ssize >= 6 then
6528:0      6      begin
6529:0      7      for index := 1 to 6 do
6530:0      7      slist[index] := ' ';
6531:0      7      ilist[1] := 6;
6532:0      7      ilist[2] := 0; ( disabled )
6533:0      7      ilist[3] := 0;
6534:0      6      end
6535:0      6      else ierr := 4;
6536:0      6      end
6537:0      6      else
6538:0      6      begin
6539:0      7      with gle_gcbl^ do
6540:0      7      if ssize >= input_name_char_count then
6541:0      8      begin
6542:0      9      for index := 1 to input_name_char_count do
6543:0      9      slist[index] := input_name[index];
6544:0      9      ilist[1] := input_name_char_count;
6545:0      9      ilist[2] := 1; ( enabled )
6546:0      9      if gle_gcbl^.input_handler_name = 'KN08 ' then
6547:0      9      ilist[3] := 255
6548:0      9      else ilist[3] := 1;
6549:0      8      end
6550:0      6      else
6551:0      6      ierr := 4;
6552:0      5      end;
6553:0      5      end;
6554:0      5
```



```
6554:C      5      end; { of case }
6555:S
6556:C      2 end;
6557:S
6558:C      1 end. { of module }
6559:S
```

No errors. No warnings.

***** Nonstandard language features enabled *****

DGL_KNOB

Description

DGL_KNOB provides DGL-level knob handler and initialization.

Requirements

DGL_TYPES, DGL_VARS, GLE_TYPES, GLE_GEN, GLE_GENI, DGL_GEN, ASM, and SYSGLOBALS.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

18069:S      4      if echo = 1 then
18070:C      4          begin
18071:C      6              with gle_knob_echo_gcb^ do
18072:C      6                  begin
18073:C      6                      info2 := 1; { do not init DGL stuff }
18074:C      6                      device_info := ADDR('3');
18075:C      6                      device_info_char_count := 1;
18076:C      6                      configure_gle ( gle_knob_echo_gcb );
18077:C      6                      if error_return <> 0 then error (err_no_display_hardware);
18078:C      6                  end;
18079:S      6              { define echo edges }
18080:C      6              input_min_x := 0;
18081:C      6              input_max_x := display_max_x;
18082:C      6              input_min_y := 0;
18083:C      6              input_max_y := display_max_y;
18084:C      6              end;
18085:C      6          end;
18086:C      4          info2 := 1;
18087:C      4      end;
18088:C      4      else
18089:C      4          begin
18090:C      4              echo_gcb := gle_gcb;
18091:C      4              with cur_disp_lim do
18092:C      4                  begin
18093:C      6                      input_min_x := trunc(xmin+0.5);
18094:C      6                      input_max_x := trunc(xmax+0.5);
18095:C      6                      input_min_y := trunc(ymin+0.5);
18096:C      6                      input_max_y := trunc(ymax+0.5);
18097:C      6                      input_cpx := d_loc_echo_x;
18098:C      6                      input_cpy := d_loc_echo_y;
18099:C      6                      info2 := display_echo_mult;
18100:C      6                  end;
18101:C      4              end;
18102:C      4          end;
18103:S      4      end;
18104:C      3      gle_start_digitize (gle_gcbi);
18105:S      3
18106:C      3      last_x := -32768;
18107:C      3      last_y := -32768;
18108:S      3
18109:C      4      repeat
18110:C      4          gle_sample ( gle_gcbi );
18111:C      4          button := gle_gcbi.info3;
18112:C      4          dx := info1;
18113:C      4          dy := info2;
18114:S      4
18115:C      4          if (dx <> last_x) or (dy <> last_y) then
18116:C      6              begin
18117:C      6                  last_x := dx;
18118:C      6                  last_y := dy;
18119:C      6                  if (echo > 0) then
18120:C      6                      with echo_gcb^ do
18121:C      7                          begin
18122:C      7                              info1 := dx;
18123:C      7                              info2 := dy;
18124:C      7                              info3 := 1;
18125:C      7                              gle_cursor ( echo_gcb );
18126:C      7                          end;
18127:C      6                  end;

```

```

18128:C      4      until button = -1;
18129:S      4
18130:C      3      gle_get_digitize(gle_gcbi);
18131:C      3      button := info3;
18132:S      3
18133:C      4      if echo > 1 then
18134:C      6          begin
18135:C      6              dx := info1;
18136:C      6              dy := info2;
18137:C      6          end;
18138:C      4      else
18139:C      4          begin
18140:C      6              last_knob_dx := dx;
18141:C      6              last_knob_dy := dy;
18142:C      6              convert_ltod(info1,info2,dx,dy);
18143:C      6          end;
18144:S      4
18145:C      3      convert_dto(dx,dy,rx,ry);
18146:C      3      adjust_return_echo ( rx, ry );
18147:S      3
18148:C      3      last_knob_rx := rx;
18149:C      3      last_knob_ry := ry;
18150:S      3
18151:C      4      if echo > 0 then
18152:C      6          with echo_gcb^ do
18153:C      6              begin
18154:C      6                  info3 := 0;
18155:C      6                  gle_cursor(echo_gcb);
18156:C      6              end;
18157:S      6          { remove cursor from screen }
18158:C      3      if echo = 1 then
18159:C      4          begin
18160:C      6              info1 := echo;
18161:C      6              gle_input_echo ( gle_gcbi );
18162:C      4          end;
18163:C      3      end;
18164:S      3
18165:C      2      if echoerror then error (err_echo_dis_int);
18166:S      2
18167:C      2      end; { await_locator }
18168:S      2
18169:D      1      procedure dgl_knob_init;
18170:S      2      begin
18171:C      2          with gcb^ do
18172:C      3              begin
18173:C      3                  proc_await_locator := knob_await_locator;
18174:C      3                  proc_sample_locator := knob_sample_locator;
18175:C      3                  last_knob_rx := 0;
18176:C      3                  last_knob_ry := 0;
18177:C      3                  last_knob_dx := 0;
18178:C      3                  last_knob_dy := 0;
18179:C      3                  last_knob_dy := 0;
18180:C      3              end;
18181:C      2          end;
18182:S      2
18183:C      1      end. { dgl_knob }
18184:S      1

```

No errors. 1 warnings.

**** Nonstandard language features enabled ****

DGL_POLY

Description

DGL_POLY contains the polygon routines except low-level code for device-dependent polygons.

Requirements

GLE_TYPES, SYSDEVS, SYSGLOBALS, GLE_HPGL_OUT, GLE_STEXT, GLE_ASTEXT, GLE_ASCLIP, GLE_SCLIP, GLE_SMARK, GLE_AUTL, GLE_UTLS, GLE_RAS_OUT, GLE_ARAS_OUT, GLE_FILE_IO, GLE_HPIB_IO, GENERAL_O, IODECLARATIONS, IOCOMASM, DGL_TOOLS, DGL_RASTER, DGL_TYPES, DGL_VARS, ASM, LOADER, GLE_GEN, DGL_GEN, DGL_HPGL, GLE_HPGL_IN, GLE_KNOB_IN, DGL_KNOB, GLE_GENI, and DGL_CONFIG_OUT.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:0      ) {
2:0      ) { Pascal work station graphics library
3:0      ) {
4:0      ) { Module = DGL_POLY
5:0      ) { Programmer = BJS
6:0      ) { Date = 11/10/82
7:0      ) {
8:0      ) { Rev history:
9:0      ) {
10:0     ) { Purpose: Hold polygon user routines
11:0     ) {
12:0     ) { (c) Copyright Hewlett-Packard Company, 1983.
13:0     ) { All rights are reserved. Copying or other
14:0     ) { reproduction of this program except for archival
15:0     ) { purposes is prohibited without the prior
16:0     ) { written consent of Hewlett-Packard Company.
17:0     ) {
18:0     ) {
19:0     ) {
20:0     ) {
21:0     ) { Use, duplication, or disclosure by the Government
22:0     ) { is subject to restrictions as set forth in
23:0     ) { paragraph (b) (3) (B) of the Rights in Technical
24:0     ) { Data and Computer Software clause in
25:0     ) { DAR 7-104.9(a).
26:0     ) {
27:0     ) { HEWLETT-PACKARD COMPANY
28:0     ) { Fort Collins, Colorado
29:0     ) {
30:0     ) { $modcal$
31:0     ) { $include 'OPTIONS'$
32:0     ) { ( This include file specifies range checking, debug and other compiler
33:0     ) { options for the graphics library
34:0     ) {
35:0     ) { $debug OFF$
36:0     ) { $range OFF$
37:0     ) { $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
38:0     ) { $FLOAT_HDW TEST$
39:0     ) {
40:0     ) {
41:0     ) {
42:0     ) {
43:0     ) { $include 'OPTIONS'$
44:0     ) { $linenum 20000$
45:0     ) { $search 'TYPES'
46:0     ) { 'DGL_VARS',
47:0     ) { 'GEN',
48:0     ) { 'GLE (18',
49:0     ) { 'LIB'$
50:0     ) {
51:0     ) {
52:0     ) {
53:0     ) {
54:0     ) {
55:0     ) {
56:0     ) {
57:0     ) {
58:0     ) {
59:0     ) {
60:0     ) {
61:0     ) {
62:0     ) {
63:0     ) {
64:0     ) {
65:0     ) {
66:0     ) {
67:0     ) {
68:0     ) {
69:0     ) {
70:0     ) {
71:0     ) {
72:0     ) {
73:0     ) {
74:0     ) {
75:0     ) {
76:0     ) {
77:0     ) {
78:0     ) {
79:0     ) {
80:0     ) {
81:0     ) {
82:0     ) {
83:0     ) {
84:0     ) {
85:0     ) {
86:0     ) {
87:0     ) {
88:0     ) {
89:0     ) {
90:0     ) {
91:0     ) {
92:0     ) {
93:0     ) {
94:0     ) {
95:0     ) {
96:0     ) {
97:0     ) {
98:0     ) {
99:0     ) {
100:0    ) {
101:0    ) {
102:0    ) {
103:0    ) {
104:0    ) {
105:0    ) {
106:0    ) {
107:0    ) {
108:0    ) {
109:0    ) {
110:0    ) {
111:0    ) {
112:0    ) {
113:0    ) {
114:0    ) {
115:0    ) {
116:0    ) {
117:0    ) {
118:0    ) {
119:0    ) {
120:0    ) {
121:0    ) {
122:0    ) {
123:0    ) {
124:0    ) {
125:0    ) {
126:0    ) {
127:0    ) {
128:0    ) {
129:0    ) {
130:0    ) {
131:0    ) {
132:0    ) {
133:0    ) {
134:0    ) {
135:0    ) {
136:0    ) {
137:0    ) {
138:0    ) {
139:0    ) {
140:0    ) {
141:0    ) {
142:0    ) {
143:0    ) {
144:0    ) {
145:0    ) {
146:0    ) {
147:0    ) {
148:0    ) {
149:0    ) {
150:0    ) {
151:0    ) {
152:0    ) {
153:0    ) {
154:0    ) {
155:0    ) {
156:0    ) {
157:0    ) {
158:0    ) {
159:0    ) {
160:0    ) {
161:0    ) {
162:0    ) {
163:0    ) {
164:0    ) {
165:0    ) {
166:0    ) {
167:0    ) {
168:0    ) {
169:0    ) {
170:0    ) {
171:0    ) {
172:0    ) {
173:0    ) {
174:0    ) {
175:0    ) {
176:0    ) {
177:0    ) {
178:0    ) {
179:0    ) {
180:0    ) {
181:0    ) {
182:0    ) {
183:0    ) {
184:0    ) {
185:0    ) {
186:0    ) {
187:0    ) {
188:0    ) {
189:0    ) {
190:0    ) {
191:0    ) {
192:0    ) {
193:0    ) {
194:0    ) {
195:0    ) {
196:0    ) {
197:0    ) {
198:0    ) {
199:0    ) {
200:0    ) {
201:0    ) {
202:0    ) {
203:0    ) {
204:0    ) {
205:0    ) {
206:0    ) {
207:0    ) {
208:0    ) {
209:0    ) {
210:0    ) {
211:0    ) {
212:0    ) {
213:0    ) {
214:0    ) {
215:0    ) {
216:0    ) {
217:0    ) {
218:0    ) {
219:0    ) {
220:0    ) {
221:0    ) {
222:0    ) {
223:0    ) {
224:0    ) {
225:0    ) {
226:0    ) {
227:0    ) {
228:0    ) {
229:0    ) {
230:0    ) {
231:0    ) {
232:0    ) {
233:0    ) {
234:0    ) {
235:0    ) {
236:0    ) {
237:0    ) {
238:0    ) {
239:0    ) {
240:0    ) {
241:0    ) {
242:0    ) {
243:0    ) {
244:0    ) {
245:0    ) {
246:0    ) {
247:0    ) {
248:0    ) {
249:0    ) {
250:0    ) {
251:0    ) {
252:0    ) {
253:0    ) {
254:0    ) {
255:0    ) {
256:0    ) {
257:0    ) {
258:0    ) {
259:0    ) {
260:0    ) {
261:0    ) {
262:0    ) {
263:0    ) {
264:0    ) {
265:0    ) {
266:0    ) {
267:0    ) {
268:0    ) {
269:0    ) {
270:0    ) {
271:0    ) {
272:0    ) {
273:0    ) {
274:0    ) {
275:0    ) {
276:0    ) {
277:0    ) {
278:0    ) {
279:0    ) {
280:0    ) {
281:0    ) {
282:0    ) {
283:0    ) {
284:0    ) {
285:0    ) {
286:0    ) {
287:0    ) {
288:0    ) {
289:0    ) {
290:0    ) {
291:0    ) {
292:0    ) {
293:0    ) {
294:0    ) {
295:0    ) {
296:0    ) {
297:0    ) {
298:0    ) {
299:0    ) {
300:0    ) {
301:0    ) {
302:0    ) {
303:0    ) {
304:0    ) {
305:0    ) {
306:0    ) {
307:0    ) {
308:0    ) {
309:0    ) {
310:0    ) {
311:0    ) {
312:0    ) {
313:0    ) {
314:0    ) {
315:0    ) {
316:0    ) {
317:0    ) {
318:0    ) {
319:0    ) {
320:0    ) {
321:0    ) {
322:0    ) {
323:0    ) {
324:0    ) {
325:0    ) {
326:0    ) {
327:0    ) {
328:0    ) {
329:0    ) {
330:0    ) {
331:0    ) {
332:0    ) {
333:0    ) {
334:0    ) {
335:0    ) {
336:0    ) {
337:0    ) {
338:0    ) {
339:0    ) {
340:0    ) {
341:0    ) {
342:0    ) {
343:0    ) {
344:0    ) {
345:0    ) {
346:0    ) {
347:0    ) {
348:0    ) {
349:0    ) {
350:0    ) {
351:0    ) {
352:0    ) {
353:0    ) {
354:0    ) {
355:0    ) {
356:0    ) {
357:0    ) {
358:0    ) {
359:0    ) {
360:0    ) {
361:0    ) {
362:0    ) {
363:0    ) {
364:0    ) {
365:0    ) {
366:0    ) {
367:0    ) {
368:0    ) {
369:0    ) {
370:0    ) {
371:0    ) {
372:0    ) {
373:0    ) {
374:0    ) {
375:0    ) {
376:0    ) {
377:0    ) {
378:0    ) {
379:0    ) {
380:0    ) {
381:0    ) {
382:0    ) {
383:0    ) {
384:0    ) {
385:0    ) {
386:0    ) {
387:0    ) {
388:0    ) {
389:0    ) {
390:0    ) {
391:0    ) {
392:0    ) {
393:0    ) {
394:0    ) {
395:0    ) {
396:0    ) {
397:0    ) {
398:0    ) {
399:0    ) {
400:0    ) {
401:0    ) {
402:0    ) {
403:0    ) {
404:0    ) {
405:0    ) {
406:0    ) {
407:0    ) {
408:0    ) {
409:0    ) {
410:0    ) {
411:0    ) {
412:0    ) {
413:0    ) {
414:0    ) {
415:0    ) {
416:0    ) {
417:0    ) {
418:0    ) {
419:0    ) {
420:0    ) {
421:0    ) {
422:0    ) {
423:0    ) {
424:0    ) {
425:0    ) {
426:0    ) {
427:0    ) {
428:0    ) {
429:0    ) {
430:0    ) {
431:0    ) {
432:0    ) {
433:0    ) {
434:0    ) {
435:0    ) {
436:0    ) {
437:0    ) {
438:0    ) {
439:0    ) {
440:0    ) {
441:0    ) {
442:0    ) {
443:0    ) {
444:0    ) {
445:0    ) {
446:0    ) {
447:0    ) {
448:0    ) {
449:0    ) {
450:0    ) {
451:0    ) {
452:0    ) {
453:0    ) {
454:0    ) {
455:0    ) {
456:0    ) {
457:0    ) {
458:0    ) {
459:0    ) {
460:0    ) {
461:0    ) {
462:0    ) {
463:0    ) {
464:0    ) {
465:0    ) {
466:0    ) {
467:0    ) {
468:0    ) {
469:0    ) {
470:0    ) {
471:0    ) {
472:0    ) {
473:0    ) {
474:0    ) {
475:0    ) {
476:0    ) {
477:0    ) {
478:0    ) {
479:0    ) {
480:0    ) {
481:0    ) {
482:0    ) {
483:0    ) {
484:0    ) {
485:0    ) {
486:0    ) {
487:0    ) {
488:0    ) {
489:0    ) {
490:0    ) {
491:0    ) {
492:0    ) {
493:0    ) {
494:0    ) {
495:0    ) {
496:0    ) {
497:0    ) {
498:0    ) {
499:0    ) {
500:0    ) {
501:0    ) {
502:0    ) {
503:0    ) {
504:0    ) {
505:0    ) {
506:0    ) {
507:0    ) {
508:0    ) {
509:0    ) {
510:0    ) {
511:0    ) {
512:0    ) {
513:0    ) {
514:0    ) {
515:0    ) {
516:0    ) {
517:0    ) {
518:0    ) {
519:0    ) {
520:0    ) {
521:0    ) {
522:0    ) {
523:0    ) {
524:0    ) {
525:0    ) {
526:0    ) {
527:0    ) {
528:0    ) {
529:0    ) {
530:0    ) {
531:0    ) {
532:0    ) {
533:0    ) {
534:0    ) {
535:0    ) {
536:0    ) {
537:0    ) {
538:0    ) {
539:0    ) {
540:0    ) {
541:0    ) {
542:0    ) {
543:0    ) {
544:0    ) {
545:0    ) {
546:0    ) {
547:0    ) {
548:0    ) {
549:0    ) {
550:0    ) {
551:0    ) {
552:0    ) {
553:0    ) {
554:0    ) {
555:0    ) {
556:0    ) {
557:0    ) {
558:0    ) {
559:0    ) {
560:0    ) {
561:0    ) {
562:0    ) {
563:0    ) {
564:0    ) {
565:0    ) {
566:0    ) {
567:0    ) {
568:0    ) {
569:0    ) {
570:0    ) {
571:0    ) {
572:0    ) {
573:0    ) {
574:0    ) {
575:0    ) {
576:0    ) {
577:0    ) {
578:0    ) {
579:0    ) {
580:0    ) {
581:0    ) {
582:0    ) {
583:0    ) {
584:0    ) {
585:0    ) {
586:0    ) {
587:0    ) {
588:0    ) {
589:0    ) {
590:0    ) {
591:0    ) {
592:0    ) {
593:0    ) {
594:0    ) {
595:0    ) {
596:0    ) {
597:0    ) {
598:0    ) {
599:0    ) {
600:0    ) {
601:0    ) {
602:0    ) {
603:0    ) {
604:0    ) {
605:0    ) {
606:0    ) {
607:0    ) {
608:0    ) {
609:0    ) {
610:0    ) {
611:0    ) {
612:0    ) {
613:0    ) {
614:0    ) {
615:0    ) {
616:0    ) {
617:0    ) {
618:0    ) {
619:0    ) {
620:0    ) {
621:0    ) {
622:0    ) {
623:0    ) {
624:0    ) {
625:0    ) {
626:0    ) {
627:0    ) {
628:0    ) {
629:0    ) {
630:0    ) {
631:0    ) {
632:0    ) {
633:0    ) {
634:0    ) {
635:0    ) {
636:0    ) {
637:0    ) {
638:0    ) {
639:0    ) {
640:0    ) {
641:0    ) {
642:0    ) {
643:0    ) {
644:0    ) {
645:0    ) {
646:0    ) {
647:0    ) {
648:0    ) {
649:0    ) {
650:0    ) {
651:0    ) {
652:0    ) {
653:0    ) {
654:0    ) {
655:0    ) {
656:0    ) {
657:0    ) {
658:0    ) {
659:0    ) {
660:0    ) {
661:0    ) {
662:0    ) {
663:0    ) {
664:0    ) {
665:0    ) {
666:0    ) {
667:0    ) {
668:0    ) {
669:0    ) {
670:0    ) {
671:0    ) {
672:0    ) {
673:0    ) {
674:0    ) {
675:0    ) {
676:0    ) {
677:0    ) {
678:0    ) {
679:0    ) {
680:0    ) {
681:0    ) {
682:0    ) {
683:0    ) {
684:0    ) {
685:0    ) {
686:0    ) {
687:0    ) {
688:0    ) {
689:0    ) {
690:0    ) {
691:0    ) {
692:0    ) {
693:0    ) {
694:0    ) {
695:0    ) {
696:0    ) {
697:0    ) {
698:0    ) {
699:0    ) {
700:0    ) {
701:0    ) {
702:0    ) {
703:0    ) {
704:0    ) {
705:0    ) {
706:0    ) {
707:0    ) {
708:0    ) {
709:0    ) {
710:0    ) {
711:0    ) {
712:0    ) {
713:0    ) {
714:0    ) {
715:0    ) {
716:0    ) {
717:0    ) {
718:0    ) {
719:0    ) {
720:0    ) {
721:0    ) {
722:0    ) {
723:0    ) {
724:0    ) {
725:0    ) {
726:0    ) {
727:0    ) {
728:0    ) {
729:0    ) {
730:0    ) {
731:0    ) {
732:0    ) {
733:0    ) {
734:0    ) {
735:0    ) {
736:0    ) {
737:0    ) {
738:0    ) {
739:0    ) {
740:0    ) {
741:0    ) {
742:0    ) {
743:0    ) {
744:0    ) {
745:0    ) {
746:0    ) {
747:0    ) {
748:0    ) {
749:0    ) {
750:0    ) {
751:0    ) {
752:0    ) {
753:0    ) {
754:0    ) {
755:0    ) {
756:0    ) {
757:0    ) {
758:0    ) {
759:0    ) {
760:0    ) {
761:0    ) {
762:0    ) {
763:0    ) {
764:0    ) {
765:0    ) {
766:0    ) {
767:0    ) {
768:0    ) {
769:0    ) {
770:0    ) {
771:0    ) {
772:0    ) {
773:0    ) {
774:0    ) {
775:0    ) {
776:0    ) {
777:0    ) {
778:0    ) {
779:0    ) {
780:0    ) {
781:0    ) {
782:0    ) {
783:0    ) {
784:0    ) {
785:0    ) {
786:0    ) {
787:0    ) {
788:0    ) {
789:0    ) {
790:0    ) {
791:0    ) {
792:0    ) {
793:0    ) {
794:0    ) {
795:0    ) {
796:0    ) {
797:0    ) {
798:0    ) {
799:0    ) {
800:0    ) {
801:0    ) {
802:0    ) {
803:0    ) {
804:0    ) {
805:0    ) {
806:0    ) {
807:0    ) {
808:0    ) {
809:0    ) {
810:0    ) {
811:0    ) {
812:0    ) {
813:0    ) {
814:0    ) {
815:0    ) {
816:0    ) {
817:0    ) {
818:0    ) {
819:0    ) {
820:0    ) {
821:0    ) {
822:0    ) {
823:0    ) {
824:0    ) {
825:0    ) {
826:0    ) {
827:0    ) {
828:0    ) {
829:0    ) {
830:0    ) {
831:0    ) {
832:0    ) {
833:0    ) {
834:0    ) {
835:0    ) {
836:0    ) {
837:0    ) {
838:0    ) {
839:0    ) {
840:0    ) {
841:0    ) {
842:0    ) {
843:0    ) {
844:0    ) {
845:0    ) {
846:0    ) {
847:0    ) {
848:0    ) {
849:0    ) {
850:0    ) {
851:0    ) {
852:0    ) {
853:0    ) {
854:0    ) {
855:0    ) {
856:0    ) {
857:0    ) {
858:0    ) {
859:0    ) {
860:0    ) {
861:0    ) {
862:0    ) {
863:0    ) {
864:0    ) {
865:0    ) {
866:0    ) {
867:0    ) {
868:0    ) {
869:0    ) {
870:0    ) {
871:0    ) {
872:0    ) {
873:0    ) {
874:0    ) {
875:0    ) {
876:0    ) {
877:0    ) {
878:0    ) {
879:0    ) {
880:0    ) {
881:0    ) {
882:0    ) {
883:0    ) {
884:0    ) {
885:0    ) {
886:0    ) {
887:0    ) {
888:0    ) {
889:0    ) {
890:0    ) {
891:0    ) {
892:0    ) {
893:0    ) {
894:0    ) {
895:0    ) {
896:0    ) {
897:0    ) {
898:0    ) {
899:0    ) {
900:0    ) {
901:0    ) {
902:0    ) {
903:0    ) {
904:0    ) {
905:0    ) {
906:0    ) {
907:0    ) {
908:0    ) {
909:0    ) {
910:0    ) {
911:0    ) {
912:0    ) {
913:0    ) {
914:0    ) {
915:0    ) {
916:0    ) {
917:0    ) {
918:0    ) {
919:0    ) {
920:0    ) {
921:0    ) {
922:0    ) {
923:0    ) {
924:0    ) {
925:0    ) {
926:0    ) {
927:0    ) {
928:0    ) {
929:0    ) {
930:0    ) {
931:0    ) {
932:0    ) {
933:0    ) {
934:0    ) {
935:0    ) {
936:0    ) {
937:0    ) {
938:0    ) {
939:0    ) {
940:0    ) {
941:0    ) {
942:0    ) {
943:0    ) {
944:0    ) {
945:0    ) {
946:0    ) {
947:0    ) {
948:0    ) {
949:0    ) {
950:0    ) {
951:0    ) {
952:0    ) {
953:0    ) {
954:0    ) {
955:0    ) {
956:0    ) {
957:0    ) {
958:0    ) {
959:0    ) {
960:0    ) {
961:0    ) {
962:0    ) {
963:0    ) {
964:0    ) {
965:0    ) {
966:0    ) {
967:0    ) {
968:0    ) {
969:0    ) {
970:0    ) {
971:0    ) {
972:0    ) {
973:0    ) {
974:0    ) {
975:0    ) {
976:0    ) {
977:0    ) {
978:0    ) {
979:0    ) {
980:0    ) {
981:0    ) {
982:0    ) {
983:0    ) {
984:0    ) {
985:0    ) {
986:0    ) {
987:0    ) {
988:0    ) {
989:0    ) {
990:0    ) {
991:0    ) {
992:0    ) {
993:0    ) {
994:0    ) {
995:0    ) {
996:0    ) {
997:0    ) {
998:0    ) {
999:0    ) {
1000:0   ) {

```

```

20016:0      ) {
20017:0      ) {
20018:0      ) {
20019:0      ) {
20020:0      ) {
20021:0      ) {
20022:0      ) {
20023:0      ) {
20024:0      ) {
20025:0      ) {
20026:0      ) {
20027:0      ) {
20028:0      ) {
20029:0      ) {
20030:0      ) {
20031:0      ) {
20032:0      ) {
20033:0      ) {
20034:0      ) {
20035:0      ) {
20036:0      ) {
20037:0      ) {
20038:0      ) {
20039:0      ) {
20040:0      ) {
20041:0      ) {
20042:0      ) {
20043:0      ) {
20044:0      ) {
20045:0      ) {
20046:0      ) {
20047:0      ) {
20048:0      ) {
20049:0      ) {
20050:0      ) {
20051:0      ) {
20052:0      ) {
20053:0      ) {
20054:0      ) {
20055:0      ) {
20056:0      ) {
20057:0      ) {
20058:0      ) {
20059:0      ) {
20060:0      ) {
20061:0      ) {
20062:0      ) {
20063:0      ) {
20064:0      ) {
20065:0      ) {
20066:0      ) {
20067:0      ) {
20068:0      ) {
20069:0      ) {
20070:0      ) {
20071:0      ) {
20072:0      ) {
20073:0      ) {
20074:0      ) {
20075:0      ) {
20076:0      ) {
20077:0      ) {
20078:0      ) {
20079:0      ) {
20080:0      ) {
20081:0      ) {
20082:0      ) {
20083:0      ) {
20084:0      ) {
20085:0      ) {
20086:0      ) {
20087:0      ) {
20088:0      ) {
20089:0      ) {
20090:0      ) {
20091:0      ) {
20092:0      ) {
20093:0      ) {
20094:0      ) {
20095:0      ) {
20096:0      ) {
20097:0      ) {
20098:0      ) {
20099:0      ) {
20100:0     ) {
20101:0     ) {
20102:0     ) {
20103:0     ) {
20104:0     ) {
20105:0     ) {
20106:0     ) {
20107:0     ) {
20108:0     ) {
20109:0     ) {
20110:0     ) {
20111:0     ) {
20112:0     ) {
20113:0     ) {
20114:0     ) {
20115:0     ) {
20116:0     ) {
20117:0     ) {
20118:0     ) {
20119:0     ) {
20120:0     ) {
20121:0     ) {
20122:0     ) {
20123:0     ) {
20124:0     ) {
20125:0     ) {
20126:0     ) {
20127:0     ) {
20128:0     ) {
20129:0     ) {
20130:0     ) {
20131:0     ) {
20132:0     ) {
20133:0     ) {
20134:0     ) {
20135:0     ) {
20136:0     ) {
20137:0     ) {
20138:0     ) {
20139:0     ) {
20140:0     ) {
20141:0     ) {
20142:0     ) {
20143:0     ) {
20144:0     ) {
20145:0     ) {
20146:0     ) {
20147:0     ) {
20148:0     ) {
20149:0     ) {
20150:0     ) {
20151:0     ) {
20152:0     ) {
20153:0     ) {
20154:0     ) {
20155:0     ) {
20156:0     ) {
20157:0     ) {
20158:0     ) {
20159:0     ) {
20160:0     ) {
20161:0     ) {
20162:0     ) {
20163:0     ) {
20164:0     ) {
20165:0     ) {
20166:0     ) {
20167:0     ) {
20168:0     ) {
20169:0     ) {
20170:0     ) {
20171:0     ) {
20172:0     ) {
20173:0     ) {
20174:0     ) {
20175:0     ) {
20176:0     ) {
20177:0     ) {
20178:0     ) {
20179:0     ) {
20180:0     ) {
20181:0     ) {
20182:0     ) {
20183:0     ) {
20184:0     ) {
20185:0     ) {
20186:0     ) {
20187:0     ) {
20188:0     ) {
20189:0     ) {
20190:0     ) {
20191:0     ) {
20192:0     ) {
20193:0     ) {
20194:0     ) {
20195:0     ) {
20196:0     ) {
20197:0     ) {
20198:0     ) {
20199:0     ) {
20200:0     ) {
20201:0     ) {
20202:0     ) {
20203:0     ) {
20204:0     ) {
20205:0     ) {
20206:0     ) {
20207:0     ) {
20208:0     ) {
20209:0     ) {
20210:0     ) {
20211:0     ) {
20212:0     ) {
20213:0     ) {
20214:0     ) {
20215:0     ) {
20216:0     ) {
20217:0     ) {
20218:0     ) {
20219:0     ) {
20220:0     ) {
20221:0     ) {
20222:0     ) {
20223:0     ) {
20224:0     ) {
20225:0     ) {
20226:0     ) {
20227:0     ) {
20228:0     ) {
20229:0     ) {
20230:0     ) {
20231:0     ) {
20232:0     ) {
20233:0     ) {
20234:0     ) {
20235:0     ) {
20236:0     ) {
20237:0     ) {
20238:0     ) {
20239:0     ) {
20240:0     ) {
20241:0     ) {
20242:0     ) {
20243:0     ) {
20244:0     ) {
20245:0     ) {
20246:0     ) {
20247:0     ) {
20248:0     ) {
20249:0     ) {
20250:0     ) {
20251:0     ) {
20252:0     ) {
20253:0     ) {
20254:0     ) {
20255:0     ) {
20256:0     ) {
20257:0     ) {
20258:0     ) {
20259:0     ) {
20260:0     ) {
20261:0     ) {
20262:0     ) {
20263:0     ) {
20264:0     ) {
20265:0     ) {
20266:0     ) {
20267:0     ) {
20268:0     ) {
20269:0     ) {
20270:0     ) {
20271:0     ) {
20272:0     ) {
20273:0     ) {
20274:0     ) {
20275:0     ) {
20276:0     ) {
20277:0     ) {
20278:0     )
```



```

20076:C      6      begin
20077:C      6      { define only one parm for dither, use lit to set }
20078:C      6      { calc brite as defined from CIE diagram }
20079:C      6      info2 := trunc((0.3*red+0.59*green+0.11*blue)*1023+0.5);
20080:C      6      info3 := 0;
20081:C      6      info4 := 0;
20082:C      6      gle_fill_index_color(gle_gcb);
20083:C      6      end
20084:C      6      else { multi color device }
20085:C      6      begin
20086:C      6      info2 := trunc(red*(23+0.5));
20087:C      6      info3 := trunc(green*1023+0.5);
20088:C      6      info4 := trunc(blue*1023+0.5);
20089:C      6      gle_fill_index_color(gle_gcb);
20090:C      6      end;
20091:C      4      end
20092:C      4      else
20093:C      4      begin
20094:C      4      info1 := 1;
20095:C      4      info2 := dgl_current_polygon_color;
20096:C      4      gle_fill_index_color(gle_gcb);
20097:C      4      end;
20098:C      3      dgl_polygon_color_current := true;
20099:C      3      end;
20100:C      2 end; { set_polygon_color }
20101:S      1 procedure set_pgn_color (index : integer);
20102:D      1
20103:S      2 { Purpose: To set the color polygons will be drawn with }
20104:D      2
20105:S      2 var
20106:D      2 pass_rgb : boolean;
20107:D      -1 2 h,s,l : real;
20108:D      -26 2
20109:S      2 begin
20110:C      2 ck_system_init;
20111:C      2 ck_display_init;
20112:C      2
20113:S      2 with gcb^,gle_gcb^ do
20114:C      2 begin { Bad values of index are set to index 1 }
20115:C      3 if (index < 0) or
20116:C      3 ((index > gamut) and
20117:C      3 ((color_table_size = 0) or (index > color_table_size))) then index := 1;
20118:C      4 dgl_current_polygon_color := index;
20119:C      3 dgl_polygon_color_current := false;
20120:C      3 end;
20121:C      3
20122:S      2 end; { set_pgn_color }
20123:C      2
20124:S      1 procedure set_pgn_ls ( index : integer);
20125:S      1
20126:D      2 { Purpose: To set the linestyle that polygons are drawn with }
20127:S      2
20128:D      2 begin
20129:C      2 ck_system_init;
20130:C      2 ck_display_init;
20131:C      2
20132:S      2 with gcb^ do
20133:C      2 begin
20134:C      2
20135:C      2

```

```

20136:C      3 if (index < 1) or (index > number_dgl_linestyles) then index := 1;
20137:C      3 dgl_current_polygon_linestyle := index;
20138:C      3 end;
20139:S      2 end; { set_pgn_line_style }
20140:C      2
20141:S      1 procedure set_pgn_style ( index : integer );
20142:D      1
20143:S      2 { Purpose: To set the polygon style that polygons will be drawn with }
20144:D      2
20145:S      2 begin
20146:C      2 ck_system_init;
20147:C      2 ck_display_init;
20148:C      2
20149:S      2 with gcb^ do
20150:C      2 begin
20151:C      3 if (index < 1) or (index > number_polygon_styles) then index := 1;
20152:C      3 with poly_table_ptr^ [ index ] do
20153:C      4 begin
20154:C      4 { decode table and setup local vars }
20155:C      4 dgl_current_polygon_crosshatch := density < 0;
20156:C      4 dgl_current_polygon_density := density;
20157:C      4 dgl_current_polygon_angle := orient;
20158:C      4 dgl_current_polygon_edge := edge;
20159:C      4 dgl_current_polygon_style := index;
20160:C      4 end;
20161:C      3 end;
20162:C      3 end;
20163:C      2 end;
20164:S      1 procedure set_pgn_table ( index : integer;
20165:D      2 pdensity : real;
20166:D      2 porient : real;
20167:D      2 pedge : integer);
20168:D      -16 2
20169:S      -16 2 { Purpose: To define an entry in the polygon table }
20170:D      -16 2
20171:S      2 begin
20172:C      2 ck_system_init;
20173:C      2 ck_display_init;
20174:C      2
20175:S      2 with gcb^ do
20176:C      2 begin
20177:C      3 begin
20178:C      3 if ((index < 1) or (index > number_polygon_styles)) or
20179:C      3 ((pedge < 0) and (pedge > 1)) or
20180:C      3 ((pdensity < -1) or (pdensity > 1)) or
20181:C      3 ((porient < -90) or (porient > 90)) then error(err_bad_parms);
20182:C      3 with poly_table_ptr^ [ index ] do
20183:C      4 begin
20184:C      4 density := pdensity;
20185:C      4 orient := porient;
20186:C      4 edge := pedge;
20187:C      4 if index = dgl_current_polygon_style then set_pgn_style(dgl_current_polygon_style);
20188:C      4 end;
20189:C      3 end;
20190:C      2 end;
20191:S      1 procedure edge_polygon ( num_points : integer;
20192:D      2 anyvar vector : gint_list;
20193:D      2 anyvar opcodes : gshortint_list;
20194:D      2 polygon_simulation : boolean );
20195:D      2

```



```

2019: S
2019: D 2 { Purpose : To draw edges around the specified polygon }
2019: S
2019: D 2 var
2020: D -4 2 vector_count : integer;
2020: D -8 next_subpolygon : integer;
2020: D -12 1 : integer;
2020: D -16 saved_color : integer;
2020: D -20 saved_linestyle : integer;
2020: D -24 saved_linewidth : integer;
2020: S
2020: C 2 begin
2020: C 2 with gcb^, gle_gcb^ do
2020: C 2 begin
2020: C 4 if (polygon_simulation) and (dgl_current_polygon_density <> 0) then
2020: C 4 begin
2020: C 4 saved_color := dgl_current_color;
2020: C 4 saved_linestyle := dgl_current_linestyle;
2020: C 4 saved_linewidth := dgl_current_linewidth;
2020: C 4 set_color(dgl_current_polygon_color);
2020: C 4 set_line_style(dgl_current_polygon_linestyle);
2020: C 4 set_line_width(1);
2020: C 4 end;
2020: S
2020: C 3 vector_count := 1;
2020: C 3 next_subpolygon := 1;
2020: S
2020: C 3 for i := 1 to num_points do
2020: C 4 begin
2020: C 4 if vector_count = next_subpolygon then
2020: C 5 begin
2020: C 5 end_x := vector[vector_count+1];
2020: C 5 end_y := vector[vector_count+2];
2020: C 5 gle_move ( gle_gcb );
2020: C 5 next_subpolygon := vector_count + vector[vector_count] * 2 + 1;
2020: C 5 vector_count := vector_count + 3;
2020: C 5 end
2020: C 5 else
2020: C 5 begin
2020: C 5 end_x := vector[vector_count];
2020: C 5 end_y := vector[vector_count+1];
2020: C 5 if opcodes[i] = 1 then gle_draw { gle_gcb };
2020: C 5 else gle_move { gle_gcb };
2020: C 5 vector_count := vector_count + 2;
2020: C 5 end;
2020: C 4 end;
2020: S
2020: C 3 if (polygon_simulation) and (dgl_current_polygon_density <> 0) then
2020: C 4 begin
2020: C 4 set_color(saved_color);
2020: C 4 set_line_style(saved_linestyle);
2020: C 4 set_line_width(saved_linewidth);
2020: C 4 end;
2020: C 3 end;
2020: S
2020: D 1 function int_div ( a, b : integer ) : integer;
2020: S
2020: D 2 { Purpose : To perform an integer div with rounding }
2020: S

```

```

2026: D -4 2 var temp : integer;
2026: S
2026: C 2 begin
2026: C 2 temp := (2 * a) div b;
2026: C 2 if odd ( temp ) then
2026: C 3 if temp > 0 then temp := temp + 1;
2026: C 4 else temp := temp - 1;
2026: C 2 int_div := temp div 2;
2026: S
2026: D 1 procedure line_line_intersection ( p1x, p1y, p2x, p2y,
2026: D 2 p3x, p3y, p4x, p4y : integer;
2026: D 2 var ix, iy : integer );
2026: S
2026: D 2 { Purpose : To calculate the intersection of two lines }
2026: D 2 { Note: The two lines must intersect. }
2026: S
2026: D 2 var
2026: D 2 num, denom,
2026: D 2 delta_x_21,
2026: D 2 delta_y_21,
2026: D 2 delta_x_31,
2026: D 2 delta_y_31,
2026: D 2 delta_x_43,
2026: D 2 delta_y_43 : integer;
2026: D -32 2 real_num,
2026: D -32 2 real_denom,
2026: D -32 2 real_factor : real;
2026: S
2026: C 2 begin
2026: C 2 $range on$
2026: C 2 delta_x_21 := p2x - p1x;
2026: C 2 delta_y_21 := p2y - p1y;
2026: C 2 delta_x_31 := p3x - p1x;
2026: C 2 delta_y_31 := p3y - p1y;
2026: C 2 delta_x_43 := p4x - p3x;
2026: C 2 delta_y_43 := p4y - p3y;
2026: C 2 try
2026: C 3 denom := delta_y_21 * delta_x_43 - delta_x_21 * delta_y_43;
2026: C 3 num := delta_x_21 * delta_y_31 - delta_y_21 * delta_x_31;
2026: C 3 ix := p3x + int_div((p4x-p3x)*num, denom);
2026: C 3 iy := p3y + int_div((p4y-p3y)*num, denom);
2026: C 3 $range off$
2026: C 3 recover
2026: C 3 if escapecode = -4 ( integer overflow ) then
2026: C 4 begin
2026: C 4 real_denom := 1.0 * delta_y_21 * delta_x_43 - 1.0 * delta_x_21 * delta_y_43;
2026: C 4 real_num := 1.0 * delta_x_21 * delta_y_31 - 1.0 * delta_y_21 * delta_x_31;
2026: C 4 real_factor := real_num / real_denom;
2026: C 4 ix := trunc(p3x + (p4x-p3x) * real_factor + 0.5);
2026: C 4 iy := trunc(p3y + (p4y-p3y) * real_factor + 0.5);
2026: C 4 end
2026: C 4 else
2026: C 4 escape(escapecode);
2026: C 2 end;

```



```

20316:S
20317:D 1 procedure draw_pg ( anyvar vector, work : gint_list;
20318:D 2 dgl_current_polygon_color,
20318:D 2 dgl_current_polygon_linesyle,
20320:D 2 normalized_sin,normalized_cos : integer;
20321:D 2 dgl_current_polygon_crosshatch : boolean;
20322:D 2 dgl_current_polygon_spacing : integer);
20323:S
20324:D 2 { PURPOSE: To draw a polygon using the current polygon attributes }
20325:S
20326:S { The input format for vector is as follows (GLE polygon format):
20327:S
20328:S VECTOR [ Number of pts in segment 1 ( 1st subpolygon ) ]
20329:S [ X1 ]
20330:S [ Y1 ]
20331:S [ X2 ]
20332:S [ Y2 ]
20333:S [ : ]
20334:S [ : ]
20335:S [ Xn ]
20336:S [ Yn ]
20337:S
20338:S [ Number of pts in segment 2 ( 2nd subpolygon ) ]
20339:S [ X1 ]
20340:S [ Y1 ]
20341:S [ : ]
20342:S [ : ]
20343:S [ Xn ]
20344:S [ Yn ]
20345:S
20346:S [ : ]
20347:S [ : ]
20348:S
20349:D 2 [ 0 ] }
20350:S
20351:S { The basic algorithm is as follow:
20352:S
20353:S - Calculate dist between fill lines
20354:S - Calculate fill line slope in terms of dx, dy
20355:S - For every edge in the polygon, calculate the x intercept
20356:S ( only intercept for x major fill slope ) along a line parallel
20357:S to the fill lines for each end point. With this information
20358:S build a record with minimum intercept, maximum intercept, and
20359:S both end points ordered by maximum intercept value.
20360:S
20361:S Maintain a minimum and maximum intercept value for all edge
20362:S end points in the polygon. This information will be used to
20363:S indicate where to start filling the polygon with fill lines.
20364:S
20365:S - Calculate using the minimum intercept value the first fill
20366:S line that may intersect the polygon.
20367:S
20368:S - For each possible fill line do the following:
20369:S
20370:S - For each intercept record look for intersections. An
20371:S intersection is determined by the current intercept value
20372:S of the current fill line, falling between the minimum
20373:S and maximum intercept values of the record.
20374:S
20375:S If an intersection is found, find the end point of

```

```

20376:S the intersection and save the point.
20377:S
20378:S After all intersections for a given fill line are found,
20379:S sort the end points. The sort alternates between top
20380:S down, and bottom up for each fill line. This minimizes
20381:S motion on mechanical devices.
20382:S
20383:D 2 Plot the end points alternating between moves and draws. )
20384:S
20385:D 2 const
20386:D 2 normal_vertex = 0;
20387:D 2 short_vertex = 1;
20388:D 2 edge_vertex = 2;
20389:S
20390:D 2 edge_index = 2;
20391:S
20392:D 2 type
20393:D 2 point_def1 = array [0..1] of integer;
20394:D 2 point_def = array [0..2] of integer;
20395:S
20396:D 2 point_array = array [1..maxint] of point_def1;
20397:S
20398:D 2 intercept_rec_def = record
20399:D 2 intercept_p_min,
20400:D 2 intercept_p_max : integer;
20401:D 2 intercept_min_points : point_def;
20402:D 2 intercept_max_points : point_def;
20403:D 2 end;
20404:S
20405:D 2 intercept_array = array [1..maxint] of intercept_rec_def;
20406:S
20407:D 2 var
20408:D 2 intercept_list_ptr : ^intercept_array;
20409:D 2 p_list_ptr : ^point_array;
20410:D 2 p_count : integer;
20411:D 2 i, j : integer;
20412:D 2 intercept_count : integer;
20413:D 2 intercept_min : integer;
20414:D 2 intercept_max : integer;
20415:D 2 intercept_inc : integer;
20416:D 2 xmin, ymin : integer;
20417:D 2 dx, dy : integer;
20418:D 2 local_spacing : integer;
20419:D 2 vector_index : integer;
20420:D 2 num_vert : integer;
20421:D 2 last_index : integer;
20422:D 2 move_it : boolean;
20423:D 2 top_down_sort : boolean;
20424:D 2 x_major : boolean;
20425:D 2 hatch : boolean;
20426:D 2 saved_color : integer;
20427:D 2 vedge_index : integer;
20428:D 2 major_index : integer;
20429:D 2 first_index : integer;
20430:D 2 nvt_edge : integer;
20431:D 2 found_fill_line_on_edge : boolean;
20432:S
20433:D 2 procedure calc_intercept ( pt_1_index : integer;
20434:D 3 pt_2_index : integer;
20435:D 3 switch_xy : boolean);

```



```

20556:D  -14 3   temp_point : integer;
20557:D  -15 3   done       : boolean;
20558:S
20559:C      3   begin
20560:C      3   if x_major then      { sort by x }
20561:C      4   begin sx := 1; sy := 0; end
20562:C      4   else
20563:C      4   begin sx := 0; sy := 1; end;
20564:S
20565:C      3   repeat
20566:C      4   index := starting + inc;
20567:C      4   done := true;
20568:C      4   test_point := p_list_ptr^[starting,sx];
20569:C      4   while index <> ending + inc do
20570:C      5   begin
20571:C      5   temp_point := p_list_ptr^[index,sx];
20572:C      5   if test_point > temp_point then
20573:C      6   begin
20574:C      6   p_list_ptr^[index,sx] := p_list_ptr^[index-inc,sx];
20575:C      6   p_list_ptr^[index-inc,sx] := temp_point;
20576:S
20577:C      6   temp_point := p_list_ptr^[index,sy];
20578:C      6   p_list_ptr^[index,sy] := p_list_ptr^[index-inc,sy];
20579:C      6   p_list_ptr^[index-inc,sy] := temp_point;
20580:S
20581:C      6   done := false;
20582:C      6   end
20583:C      6   else
20584:C      6   test_point := temp_point;
20585:C      6   index := index + inc;
20586:C      5   end;
20587:C      4   until done { sort }
20588:C      4   end;
20589:S
20590:C      2   begin { poly }
20591:S
20592:C      2   with gcb^ do
20593:C      3   begin
20594:C      3   intercept_list_ptr := addr(work);
20595:C      3   hatch := dgl_current_polygon_crosshatch;
20596:C      3   local_spacing := dgl_current_polygon_spacing;
20597:S
20598:C      3   saved_color := dgl_current_color;
20599:C      3   if dgl_current_color <> dgl_current_polygon_color then set_color(dgl_current_polygon_color);
20600:S
20601:C      3   with gle_gcb^ do
20602:C      4   repeat { cross hatching loop }
20603:S
20604:C      5   hatch := not hatch;
20605:S
20606:C      5   { Calc slope in terms of dx, dy }
20607:C      5   { Calc x or y spacing (intercept_inc) }
20608:S
20609:C      5   x_major := true;
20610:C      5   major_index := 1;
20611:C      5   if abs(normalized_sin) = normalized_one { 90 deg } then
20612:C      6   begin
20613:C      6   dy := display_max_y;
20614:C      6   dx := 0;
20615:C      6   intercept_inc := local_spacing;

```

```

20616:C      6   end
20617:C      6   else
20618:C      6   if abs(normalized_sin) <= abs(normalized_cos) { <= 45 deg } then
20619:C      7   begin
20620:C      7   dx := display_max_x;
20621:C      7   dy := int_div(dx * normalized_sin, normalized_cos);
20622:C      7   x_major := false;
20623:C      7   major_index := 0;
20624:C      7   intercept_inc := abs(int_div(local_spacing * normalized_one, normalized_cos));
20625:C      7   end
20626:C      7   else
20627:C      7   begin
20628:C      7   dy := display_max_y;
20629:C      7   dx := int_div(dy * normalized_cos, normalized_sin);
20630:C      7   intercept_inc := abs(int_div(local_spacing * normalized_one, normalized_sin));
20631:C      7   end;
20632:S
20633:C      5   if intercept_inc < 1 then intercept_inc := 1;
20634:S
20635:C      5   { Calc end point intercepts }
20636:S
20637:C      5   intercept_count := 1;
20638:C      5   intercept_min := maxint;
20639:C      5   intercept_max := minint;
20640:S
20641:C      5   vector_index := 1;
20642:C      5   while vector_index <> 0 do
20643:C      6   begin
20644:C      6   num_vert := vector_index;
20645:C      6   vector_index := vector_index + 1;
20646:C      6   for i := 2 to num_vert do
20647:C      7   calc_intercept ( vector_index+(i-1)*2, vector_index+(i-2)*2, x_major);
20648:C      7   last_index := vector_index+(num_vert-1)*2;
20649:C      7   calc_intercept ( vector_index, last_index, x_major);
20650:C      6   vector_index := last_index + 2;
20651:C      6   end;
20652:S
20653:C      5   intercept_count := intercept_count - 1;
20654:S
20655:C      5   vector_index := 1;
20656:C      5   vedge_index := 1;
20657:C      5   while vector_index <> 0 do
20658:C      6   begin
20659:C      6   num_vert := vector_index;
20660:C      6   first_index := vedge_index;
20661:C      6   last_index := vedge_index + num_vert;
20662:C      6   for i := 1 to num_vert do
20663:C      7   begin
20664:C      7   next_edge := vedge_index + 1;
20665:C      7   { The following while statement should read }
20666:C      7   { 'while (next_edge < last_index) and ...' }
20667:C      7   { however this "bug" was not found until after }
20668:C      7   { QA. It will not procedure a user bug though }
20669:C      7   { since the following 'if' stmt with not use the }
20670:C      7   { bad results }
20671:C      7   while (next_edge < last_index) and
20672:C      8   (intercept_list_ptr^[next_edge].
20673:C      8   intercept_max_points[edge_index] = edge_vertex) do
20674:C      8   next_edge := next_edge + 1;
20675:C      7   if next_edge >= last_index then

```



```

20676:C      8      begin
20677:C      8      next_edge := first_index;
20678:C      8      while (next_edge < vedge_index) and
20679:C      9      (intercept_list_ptr^[next_edge].
20680:C      9      intercept_max_points[edge_index] = edge_vertex) do
20681:C      9      next_edge := next_edge + 1;
20682:C      7      end;
20683:C      7      if (intercept_list_ptr^[next_edge].
20684:C      8      intercept_max_points[edge_index] <> edge_vertex) then
20685:C      8      calc_vertex_info ( vedge_index,next_edge);
20686:C      7      vedge_index := vedge_index + 1;
20687:C      7      end;
20688:C      6      vector_index := vector_index + num_vert * 2 + 1;
20689:C      6      end;
20690:C      5      { Calc first fill line intercept value, adjust with lower left }
20691:C      5      of display
20692:C      5      p_list_ptr := addr(work,(intercept_count+1)*32);
20693:C      5      intercept_min := intercept_min - (intercept_min mod intercept_inc);
20694:C      5      xmin := 0;
20695:C      5      ymin := 0;
20696:C      5      top_down_sort := true;
20697:C      5      { Fill polygon }
20698:C      5      while intercept_min <= intercept_max do
20699:C      6      begin
20700:C      6      if x_major then
20701:C      7      xmin := intercept_min
20702:C      7      else
20703:C      7      ymin := intercept_min;
20704:C      6      { Find intersections }
20705:C      6      p_count := 0;
20706:C      6      found_fill_line_on_edge := false;
20707:C      6      for i := 1 to intercept_count do
20708:C      7      begin
20709:C      7      with intercept_list_ptr^[i] do
20710:C      8      begin
20711:C      8      if (intercept_min = intercept_p_min) and
20712:C      9      (intercept_min = intercept_p_max) then
20713:C      9      found_fill_line_on_edge := true
20714:C      9      else
20715:C      9      if (intercept_min >= intercept_p_min) and
20716:C      10     (intercept_min <= intercept_p_max) then
20717:C      10     { intersection }
20718:C      10     begin
20719:C      10     if (intercept_min = intercept_p_min) then
20720:C      11     begin
20721:C      11     if (intercept_min_points[edge_index] = normal_vertex) then
20722:C      12     begin
20723:C      12     p_count := p_count + 1;
20724:C      12     p_list_ptr^[p_count,0] := intercept_min_points[0];
20725:C      12     p_list_ptr^[p_count,1] := intercept_min_points[1];
20726:C      12     end;
20727:C      11     end
20728:C      11     else
20729:C      11     if (intercept_min = intercept_p_max) then
20730:C      12     begin
20731:C      12     if (intercept_max_points[edge_index] = normal_vertex) then
20732:C      13     begin
20733:C      13     p_count := p_count + 1;
20734:C      13     p_list_ptr^[p_count,0] := intercept_max_points[0];
20735:C      13     p_list_ptr^[p_count,1] := intercept_max_points[1];
20736:C      13     end;
20737:C      12     end
20738:C      11     end
20739:C      10     end
20740:C      10     end
20741:C      10     end
20742:C      10     end
20743:C      10     end
20744:C      10     end
20745:C      10     end
20746:C      10     end
20747:C      10     end
20748:C      10     end
20749:C      10     end
20750:C      10     end
20751:C      10     end
20752:C      10     end
20753:C      10     end
20754:C      10     end
20755:C      10     end
20756:C      10     end
20757:C      10     end
20758:C      10     end
20759:C      10     end
20760:C      10     end
20761:C      10     end
20762:C      10     end
20763:C      10     end
20764:C      10     end
20765:C      10     end
20766:C      10     end
20767:C      10     end
20768:C      10     end
20769:C      10     end
20770:C      10     end
20771:C      10     end
20772:C      10     end
20773:C      10     end
20774:C      10     end
20775:C      10     end
20776:C      10     end
20777:C      10     end
20778:C      10     end
20779:C      10     end
20780:C      10     end
20781:C      10     end
20782:C      10     end
20783:C      10     end
20784:C      10     end
20785:C      10     end
20786:C      10     end
20787:C      10     end
20788:C      10     end
20789:C      10     end
20790:C      10     end
20791:C      10     end
20792:C      10     end
20793:C      10     end
20794:C      10     end
20795:C      10     end

```

```

20736:C      12     if (intercept_max_points[edge_index] = normal_vertex) then
20737:C      13     begin
20738:C      13     p_count := p_count + 1;
20739:C      13     p_list_ptr^[p_count,0] := intercept_max_points[0];
20740:C      13     p_list_ptr^[p_count,1] := intercept_max_points[1];
20741:C      13     end;
20742:C      12     end;
20743:C      12     else
20744:C      12     begin
20745:C      12     p_count := p_count + 1;
20746:C      12     line_line_intersection(xmin,ymin,xmin+dx,ymin+dy,
20747:C      12     intercept_min_points[0],intercept_min_points[1],
20748:C      12     intercept_max_points[0],intercept_max_points[1],
20749:C      12     p_list_ptr^[p_count,0],p_list_ptr^[p_count,1]);
20750:C      12     end;
20751:C      10     end;
20752:C      10     end;
20753:C      7     end;
20754:C      6     if found_fill_line_on_edge then { add edge points }
20755:C      7     begin
20756:C      7     if p_count > 1 then sort(1,p_count,1); { sort bottom up }
20757:C      7     if odd(p_count) then p_count := p_count-1; { remove last move }
20758:C      7     for i := 1 to intercept_count do
20759:C      8     with intercept_list_ptr^[i] do
20760:C      9     begin
20761:C      9     if (intercept_min = intercept_p_min) and
20762:C      10     (intercept_min = intercept_p_max) then
20763:C      10     begin
20764:C      10     p_count := p_count + 1;
20765:C      10     p_list_ptr^[p_count,0] := intercept_min_points[0];
20766:C      10     p_list_ptr^[p_count,1] := intercept_min_points[1];
20767:C      10     p_count := p_count + 1;
20768:C      10     p_list_ptr^[p_count,0] := intercept_max_points[0];
20769:C      10     p_list_ptr^[p_count,1] := intercept_max_points[1];
20770:C      10     end;
20771:C      9     end;
20772:C      9     end;
20773:C      7     i := 1;
20774:C      7     repeat
20775:C      8     j := i + 2;
20776:C      8     while j < p_count do
20777:C      9     begin
20778:C      9     if (p_list_ptr^[i,major_index] <=
20779:C      10     p_list_ptr^[j+1,major_index]) and
20780:C      10     (p_list_ptr^[j,major_index] <=
20781:C      10     p_list_ptr^[i+1,major_index]) then
20782:C      10     begin
20783:C      10     if p_list_ptr^[i,major_index] >
20784:C      11     p_list_ptr^[j,major_index] then
20785:C      11     begin
20786:C      11     p_list_ptr^[i,0] := p_list_ptr^[j,0];
20787:C      11     p_list_ptr^[i,1] := p_list_ptr^[j,1];
20788:C      11     end;
20789:C      10     if p_list_ptr^[i+1,major_index] <
20790:C      11     p_list_ptr^[j+1,major_index] then
20791:C      11     begin
20792:C      11     p_list_ptr^[i+1,0] := p_list_ptr^[j+1,0];
20793:C      11     p_list_ptr^[i+1,1] := p_list_ptr^[j+1,1];
20794:C      11     end;
20795:C      10     for t := j to p_count-2 do

```



```

20796:C      11      begin
20797:C      11      p_list_ptr^[t,0] := p_list_ptr^[t+2,0];
20798:C      11      p_list_ptr^[t,1] := p_list_ptr^[t+2,1];
20799:C      11      end;
20800:C      10      p_count := p_count - 2;
20801:C      10      j := i + 2;
20802:C      10      end
20803:C      10      else
20804:C      10      j := j + 2;
20805:C      10      end;
20806:C      8      i := i + 2;
20807:C      8      until i > p_count;
20808:C      7      end;
20809:S
20810:C      6      { Sort points }
20811:C      6      if p_count > 1 then
20812:C      7      begin
20813:C      7      top_down_sort := not top_down_sort;
20814:C      7      if top_down_sort then sort(p_count,1,-1)
20815:C      8      else sort(1,p_count,1);
20816:C      7      end;
20817:S
20818:C      6      { draw a fill line }
20819:C      6      move_it := true;
20820:C      6      for i := 1 to p_count do
20821:C      7      begin
20822:C      7      end_x := p_list_ptr^[i,0];
20823:C      7      end_y := p_list_ptr^[i,1];
20824:C      7      if move_it then call { move,gle_gcb }
20825:C      8      else call { draw,gle_gcb };
20826:C      7      move_it := not move_it;
20827:C      7      end;
20828:S
20829:C      6      intercept_min := intercept_min + intercept_inc;
20830:C      6      end; { of filling loop }
20831:S
20832:C      5      if (not hatch) and dgl_current_polygon_crosshatch then
20833:C      6      begin
20834:C      6      t := normalized_sin;
20835:C      6      normalized_sin := -normalized_cos;
20836:C      6      normalized_cos := t;
20837:C      6      end;
20838:C      5      until hatch; { end of hatching loop }
20839:S
20840:C      3      if saved_color <> dgl_current_polygon_color then set_color(saved_color);
20841:C      3      end;
20842:C      2 end;
20843:S
20844:D      1 procedure draw_polygon ( anyvar vector, work : gint_list );
20845:S
20846:D      2 { PURPOSE: To draw a polygon using the current polygon attributes }
20847:S
20848:D      { The input format for vector is as follows (GLE polygon format):
20849:S
20850:D      VECTOR [ Number of pts in segment 1 ( 1st subpolygon ) ]
20851:D      [ X1 ]
20852:D      [ Y1 ]
20853:D      [ X2 ]
20854:D      [ Y2 ]
20855:D      [ : ]

```

```

20856:D      [ Xn ]
20857:D      [ Yn ]
20858:D      [ : ]
20859:D      [ Number of pts in segment 2 ( 2nd subpolygon ) ]
20860:D      [ X1 ]
20861:D      [ Y1 ]
20862:D      [ : ]
20863:D      [ Xn ]
20864:D      [ Yn ]
20865:D      [ : ]
20866:D      [ Xm ]
20867:D      [ Ym ]
20868:D      [ : ]
20869:D      [ : ]
20870:D      [ : ]
20871:D      2      [ 0 ]
20872:D
20873:D      2 { normalized_one = 32768 }
20874:S
20875:D      2 var
20876:D      -8 local_angle : real;
20877:D      -9 hatch : boolean;
20878:D      -14 local_spacing : integer;
20879:D      -22 rad_angle : real;
20880:D      -22 sin_angle,
20881:D      -30 cos_angle : integer { normalized fixed point numbers };
20882:S
20883:D      2 begin { draw_polygon }
20884:S
20885:C      2 with gcb^ do
20886:C      3 begin
20887:C      3 if dgl_current_polygon_density <> 0 then
20888:C      4 begin
20889:C      4 local_angle := dgl_current_polygon_angle;
20890:C      4 hatch := dgl_current_polygon_crosshatch and (dgl_current_polygon_density <> 1);
20891:C      4 if dgl_current_polygon_density = 1 then
20892:C      5 local_spacing := gle_gcb^.polygon_solid_fill
20893:C      5 else
20894:C      5 local_spacing :=
20895:C      5 abs(trunc(1/dgl_current_polygon_density * gle_gcb^.polygon_fill_factor));
20896:C      4 if local_spacing < 1 then local_spacing := 1;
20897:C      4 if hatch then local_spacing := local_spacing * 2;
20898:C      4 if local_spacing = 1 then local_angle := 0;
20899:C      4 rad_angle := deg_to_rad * local_angle;
20900:C      4 sin_angle := trunc(sin(rad_angle) * normalized_one);
20901:C      4 cos_angle := trunc(cos(rad_angle) * normalized_one);
20902:C      4 draw_pg(vector,work,dgl_current_polygon_color,dgl_current_polygon_linestyle,
20903:C      4 sin_angle,cos_angle,hatch,local_spacing);
20904:C      4 end;
20905:C      3 end;
20906:C      2 end;
20907:S
20908:D      1 procedure setup_for_polygon ( real_format : boolean;
20909:D      2 num_points : integer;
20910:D      2 anyvar xvec, yvec : gshortint_list;
20911:D      2 anyvar rxvec, ryvec : greal_list;
20912:D      2 anyvar opcodes : gshortint_list;
20913:D      2 anyvar vector : vec_ptr_def;
20914:D      2 work_mult : integer;
20915:D      2 anyvar work_ptr : work_ptr_def;

```



```

20918:D      2      var last_subpoly : integer ;;
20919:S
20918:D      { Purpose : To prepare for drawing a polygon set. This includes
20919:D      { setting up attributes, creating work space, performing error
20920:D      { checks, and creating a GLE format polygon.
20921:S
20922:D      2      var
20923:D      2      1      : integer;
20924:D      2      sub_poly_start : integer;
20925:D      2      sub_poly_count : integer;
20926:D      2      point_count : integer;
20927:D      2      vector_count : integer;
20928:D      2      local_angle : real;
20929:D      2      rad_angle : real;
20930:S
20931:D      2      begin
20932:D      2      ck_system_init;
20933:D      2      ck_display_init;
20934:D      2      $ovflcheck on$
20935:D      2      if num_points <= 0 then error (err_neg_points);
20936:D      2      if opcodes[1] <> 2 then error (err_bad_parms);
20937:S
20938:D      2      { allocate worst possible space for vector array }
20939:D      2      mark(stack_ptr); { mark current base }
20940:D      2      newbytes(vector,12*num_points+4); { allocate worst case space }
20941:S
20942:D      2      with gcb^ do
20943:D      2      begin
20944:D      2      saved_linestyle := dgl_current_linestyle;
20945:D      2      saved_linewidth := dgl_current_linewidth;
20946:S
20947:D      2      if dgl_current_linestyle <> dgl_current_polygon_linestyle then
20948:D      2      set_line_style ( dgl_current_polygon_linestyle );
20949:D      2      if dgl_current_linewidth <> 1 then set_line_width(1);
20950:D      2      if not dgl_polygon_color_current then set_polygon_color;
20951:D      2      end;
20952:S
20953:D      2      sub_poly_start := 1;
20954:D      2      i := 1;
20955:D      2      sub_poly_count := 1;
20956:D      2      point_count := 1;
20957:D      2      last_subpoly := 1; { last subpolygon in polygon }
20958:D      2      vector_count := 2; { first spot will hold count }
20959:S
20960:D      2      while i <= num_points do
20961:D      2      begin
20962:D      2      if (opcodes[i]=2) and (i<>1) then
20963:D      2      begin
20964:D      2      vector[sub_poly_start] := point_count-1;
20965:D      2      sub_poly_start := vector_count; { save space to hold count }
20966:D      2      vector_count := vector_count + 1;
20967:D      2      point_count := 1;
20968:D      2      sub_poly_count := sub_poly_count + 1;
20969:D      2      last_subpoly := i;
20970:D      2      end;
20971:D      2      point_count := point_count + 1;
20972:D      2      if real_format then
20973:D      2      convert_wtod(xvec[i],yvec[i],vector[vector_count],vector[vector_count+1])
20974:D      2      else
20975:D      2      if short_flag then

```

```

20976:D      5      convert_intwtod(xvec[i],yvec[i],vector[vector_count],vector[vector_count+1])
20977:D      5      else
20978:D      5      convert_wtod(xvec[i],yvec[i],vector[vector_count],vector[vector_count+1]);
20979:D      5      vector_count := vector_count + 2;
20980:D      5      i := i + 1;
20981:D      5      end;
20982:D      2      vector[sub_poly_start] := point_count-1;
20983:D      2      vector[vector_count] := 0;
20984:S
20985:D      2      newbytes(work_ptr,work_mult*vector_count); { allocate work space for gle }
20986:S
20987:D      2      with gcb^,gle_gcb^ do
20988:D      2      begin
20989:D      2      if dgl_current_polygon_crosshatch then info1 := 1;
20990:D      2      else info1 := 0;
20991:S
20992:D      2      local_angle := dgl_current_polygon_angle;
20993:D      2      rad_angle := deg_to_rad * local_angle;
20994:D      2      info3 := trunc(sin(rad_angle) * normalized_one);
20995:D      2      info4 := trunc(cos(rad_angle) * normalized_one);
20996:S
20997:D      2      if dgl_current_polygon_density = 1 then
20998:D      2      info2 := polygon_solid_fill;
20999:D      2      else
21000:D      2      if dgl_current_polygon_density = 0 then
21001:D      2      info2 := 0;
21002:D      2      else
21003:D      2      begin
21004:D      2      info2 := abs(trunc(1/dgl_current_polygon_density * polygon_fill_factor));
21005:D      2      if info2 < 1 then info2 := 1;
21006:D      2      end;
21007:D      2      end;
21008:S
21009:D      2      end;
21010:S
21011:D      1      procedure finish_polygon;
21012:D      2      { Purpose: Restore linestyle and line width to current values }
21013:D      2      begin
21014:D      2      with gcb^ do
21015:D      2      begin
21016:D      2      if saved_linestyle <> dgl_current_polygon_linestyle then
21017:D      2      set_line_style(saved_linestyle);
21018:D      2      if saved_linewidth <> 1 then
21019:D      2      set_line_width(saved_linewidth);
21020:D      2      end;
21021:D      2      end;
21022:S
21023:D      1      procedure int_polygon_dd ( num_points : integer;
21024:D      2      anyvar xvec, yvec : gshortint_list;
21025:D      2      anyvar opcodes : gshortint_list );
21026:S
21027:D      2      { Purpose : To output a device dependent polygon set }
21028:S
21029:D      2      var
21030:D      2      1      : array[1..1] of real;
21031:D      2      work_ptr : work_ptr_def;
21032:D      2      vector_ptr : vec_ptr_def;
21033:D      2      last_subpolygon : integer;

```



```

21036:D -21 2 use_simulation : boolean;
21037:S
21038:C
21039:C 2 begin
21040:C 3 try ( must return 'new' space if escape occurs )
21041:C 3 setup_for_polygon ( false, num_points, xvec, yvec, t, t,
21042:S 3 opcodes, vector_ptr, 14, work_ptr, last_subpolygon);
21043:C 3 with gle_gcb^ do
21044:C 4 begin
21045:C 4 use_simulation := true;
21046:C 4 if polygon_support = 1 then
21047:C 5 begin
21048:C 5 gle_get_polygon_info ( gle_gcb );
21049:C 5 if error_return = 0 then
21050:C 6 begin
21051:C 6 info_ptr1 := vector_ptr;
21052:C 6 info_ptr2 := work_ptr;
21053:C 6 gle_polygon ( gle_gcb );
21054:C 6 use_simulation := false;
21055:C 6 end;
21056:C 5 end;
21057:C 4 end;
21058:C 3 finish_polygon;
21059:C 3 if gcb^.dgl_current_polygon_edge or (use_simulation) then
21060:C 4 edge_polygon ( num_points, vector_ptr^ [1], opcodes, use_simulation);
21061:C 4 int_move(xvec[last_subpolygon], yvec[last_subpolygon]); { update cp }
21062:C 3
21063:C 3 release(stack_ptr); { return all space }
21064:C 3 recover
21065:C 3 begin
21066:C 3 if escapecode <> -27 then release (stack_ptr);
21067:C 3 escape(escapecode);
21068:C 3 end;
21069:C 2 end;
21070:S
21071:D 1 procedure polygon_dev_dep ( num_points : integer;
21072:D 2 anyvar xvec, yvec : greal_list;
21073:D 2 anyvar opcodes : gshortint_list );
21074:S
21075:D 2 { Purpose : To output a device dependent polygon set }
21076:S
21077:D 2 var
21078:D -8 2 t : array[1..1] of real;
21079:D -12 2 work_ptr : work_ptr_def;
21080:D -16 2 vector_ptr : vec_ptr_def;
21081:D -20 2 last_subpolygon : integer;
21082:D -21 2 use_simulation : boolean;
21083:S
21084:C 2 begin
21085:C 3 try ( must return 'new' space if escape occurs )
21086:C 3 setup_for_polygon ( true, num_points, t, t, xvec, yvec,
21087:C 3 opcodes, vector_ptr, 14, work_ptr, last_subpolygon);
21088:C 3 with gle_gcb^ do
21089:C 4 begin
21090:C 4 use_simulation := true;
21091:C 4 if polygon_support = 1 then
21092:C 5 begin
21093:C 5 gle_get_polygon_info ( gle_gcb );
21094:C 5 if error_return = 0 then
21095:C 6 begin

```

```

21096:C 6 info_ptr1 := vector_ptr;
21097:C 6 info_ptr2 := work_ptr;
21098:C 6 gle_polygon ( gle_gcb );
21099:C 6 use_simulation := false;
21100:C 5 end;
21101:C 4 end;
21102:C 3 finish_polygon;
21103:C 3 if gcb^.dgl_current_polygon_edge or (use_simulation) then
21104:C 4 edge_polygon ( num_points, vector_ptr^ [1], opcodes, use_simulation);
21105:C 4 move(xvec[last_subpolygon], yvec[last_subpolygon]); { update cp }
21106:C 3
21107:C 3 release(stack_ptr); { return all space }
21108:C 3 recover
21109:C 3 begin
21110:C 3 if escapecode <> -27 then release (stack_ptr);
21111:C 3 escape(escapecode);
21112:C 3 end;
21113:C 2 end;
21114:S
21115:D 1 procedure int_polygon ( num_points : integer; anyvar xvec, yvec : gshortint_list;
21116:D 2 anyvar opcodes : gshortint_list );
21117:D 2
21118:S
21119:D 2 { Purpose : To output a device independent polygon set }
21120:D 2
21121:S
21122:D 2 var
21123:D -8 2 t : array[1..1] of real;
21124:D -12 2 work_ptr : work_ptr_def;
21125:D -16 2 vector_ptr : vec_ptr_def;
21126:D -20 2 last_subpolygon : integer;
21127:S
21128:C 2 begin
21129:C 3 try ( must return 'new' space if escape occurs )
21130:C 3 setup_for_polygon ( false, num_points, xvec, yvec, t, t,
21131:C 3 opcodes, vector_ptr, 40, work_ptr, last_subpolygon );
21132:C 3
21133:C 3 draw_polygon ( vector_ptr^ [1], work_ptr^ [1] );
21134:C 3 finish_polygon;
21135:C 3 if gcb^.dgl_current_polygon_edge then
21136:C 4 edge_polygon ( num_points, vector_ptr^ [1], opcodes, false );
21137:C 4 int_move(xvec[last_subpolygon], yvec[last_subpolygon]); { update cp }
21138:C 3
21139:C 3 release(stack_ptr); { return all space }
21140:C 3 recover
21141:C 3 begin
21142:C 3 if escapecode <> -27 then release (stack_ptr);
21143:C 3 escape(escapecode);
21144:C 3 end;
21145:C 2 end;
21146:S
21147:D 1 procedure polygon ( num_points : integer;
21148:D 2 anyvar xvec, yvec : greal_list;
21149:D 2 anyvar opcodes : gshortint_list );
21150:S
21151:D 2 { Purpose : To output a device independent polygon set }
21152:D 2
21153:S
21154:D 2 var
21155:D -8 2 t : array[1..1] of real;
21156:D -12 2 work_ptr : work_ptr_def;

```



```
21155:D -18 2 vector_ptr : vec_ptr_def;
21157:D -20 2 last_subpolygon : integer;
21159:S
21163:C 2 begin
21160:C 2 try { must return 'new' space if escape occurs }
21161:C 3 setup_for_polygon ( true, num_points, t, t, xvec, yvec,
21162:C 3 opcodes, vector_ptr,40, work_ptr,last_subpolygon );
21163:S
21164:C 3 draw_polygon ( vector_ptr^ [1], work_ptr^ [1] );
21165:C 3 finish_polygon;
21166:C 3 if gcb.dgl_current_polygon_edge then
21167:C 4 edge_polygon ( num_points, vector_ptr^ [1], opcodes, false );
21168:C 3 move(xvec[last_subpolygon],yvec[last_subpolygon]); { update cp }
21169:S
21170:C 3 release(stack_ptr); { return all space }
21171:C 3 recover
21172:C 3 begin
21173:C 3 if escapecode <> -27 then release (stack_ptr);
21174:C 3 escape(escapecode);
21175:C 3 end;
21176:C 2 end;
21177:S
21178:C 1 end. (module DGL_POLY)
21179:S
21180:D 1
21181:S
```

No errors. No warnings.

***** Nonstandard language features enabled *****

DGL_RAS

Description

DGL_RAS provides device-dependent initialization for raster devices, as well as raster utilities.

Requirements

DGL_TYPES, DGL_VARS, SYSDEVS, SYSGLOBALS, ASM, LOADERS, GLE_GEN, GLE_TYPES, GLE_GEN, GLE_AUTL, GLE_RAS_OUT, and DGL_GEN.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:0      0:
2:0      0: DGL device dependent init routine
3:0      0:
4:0      0: Module = DGL_RASTER
5:0      0: Programmer = BJS
6:0      0: Date = 1 - 5-83
7:0      0:
8:0      0: Purpose: To provide device dependent initialization for raster devices.
9:0      0:
10:0     0: Rev history
11:0     0: Created - 1 - 5-83 BJS
12:0     0: Modified - 02-17-84 BDS Changed allocations from dynamic to global.
13:0     0:
14:0     0: (c) Copyright Hewlett-Packard Company, 1983.
15:0     0: All rights are reserved. Copying or other
16:0     0: reproduction of this program except for archival
17:0     0: purposes is prohibited without the prior
18:0     0: written consent of Hewlett-Packard Company.
19:0     0:
20:0     0:
21:0     0:
22:0     0:
23:0     0: RESTRICTED RIGHTS LEGEND
24:0     0: Use, duplication, or disclosure by the Government
25:0     0: is subject to restrictions as set forth in
26:0     0: paragraph (b) (3) (B) of the Rights in Technical
27:0     0: Data and Computer Software clause in
28:0     0: DAR 7-104.9(a).
29:0     0:
30:0     0: HEWLETT-PACKARD COMPANY
31:0     0: Fort Collins, Colorado
32:0     0:
33:0     0: $search 'GLE LIB',
34:0     0: $types
35:0     0: 'DGL_VAS',
36:0     0: 'GEN'S
37:0     0: $modals
38:0     0: $include 'OPTIONS'S { ***** COMPILER OPTIONS ***** }
39:0     0: { This include file specifies range checking, debug and other compiler
40:0     0: options for the graphics library
41:0     0:
42:0     0: $debug OFF$
43:0     0: $range OFF$
44:0     0: $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'S
45:0     0: $FLOAT_HDW TEST$
46:0     0:
47:0     0:
48:0     0:
49:0     0:
50:0     0: $include 'OPTIONS'S { ***** COMPILER OPTIONS ***** }
51:0     0: $linenum 1700$
52:0     0:
53:0     0: module DGL_RASTER;
54:0     0:
55:0     0: 1 export
56:0     0:
57:0     0: 1 procedure dgl_raster_init ( control : integer );
58:0     0:
59:0     0: 1 implement
60:0     0:

```

```

17010 0 1 import dgl_types,
17011 0 1 dgl_vars,
17012 0 1 sysdevs,
17013 0 1 asm,
17014 0 1 sysglobals,
17015 0 1 gle_types,
17016 0 1 gle_gen,
17017 0 1 gle_autl,
17018 0 1 gle_ras_out,
17019 0 1 dgl_gen;
17020 0 1
17021 0 1 type
17022 0 1 init_color_table_def = ARRAY [0..15] of c_def;
17023 0 1
17024 0 1 const
17025 0 1 init_color_table = init_color_table_def [
17026 0 1 c_def [ red : 0, green : 0, blue : 0 ], { 0 }
17027 0 1 c_def [ red : 1, green : 1, blue : 1 ], { 1 }
17028 0 1 c_def [ red : 1, green : 0, blue : 0 ], { 2 }
17029 0 1 c_def [ red : 1, green : 1, blue : 0 ], { 3 }
17030 0 1 c_def [ red : 0, green : 1, blue : 0 ], { 4 }
17031 0 1 c_def [ red : 0, green : 0, blue : 1 ], { 5 }
17032 0 1 c_def [ red : 0, green : 0, blue : 1 ], { 6 }
17033 0 1 c_def [ red : 1, green : 0, blue : 1 ], { 7 }
17034 0 1 c_def [ red : 0, green : 0, blue : 0 ], { 8 }
17035 0 1 c_def [ red : 0.8, green : 0.73333333333333, blue : 0.2 ], { 9 }
17036 0 1 c_def [ red : 0.2, green : 0.66666666666667, blue : 0.46666666666667 ], { 10 }
17037 0 1 c_def [ red : 0.53333333333333, green : 0.4, blue : 0.66666666666667 ], { 11 }
17038 0 1 c_def [ red : 0.8, green : 0.26666666666667, blue : 0.4 ], { 12 }
17039 0 1 c_def [ red : 1.0, green : 0.4, blue : 0.2 ], { 13 }
17040 0 1 c_def [ red : 1.0, green : 0.46666666666667, blue : 0 ], { 14 }
17041 0 1 c_def [ red : 0.86666666666667, green : 0.53333333333333, blue : 0.26666666666667 ], { 15 }
17042 0 1
17043 0 1 procedure hpm_new(var object:anyptr; numbytes : integer); external;
17044 0 1
17045 0 1 procedure raster_linestyle ( index : integer);
17046 0 1
17047 0 2 { Purpose: To set the linestyle that primitives are drawn with
17048 0 2 }
17049 0 2 type
17050 0 2 ls_map_def = packed array [1..13] of gbyte;
17051 0 2 const
17052 0 2 ls_map = ls_map_def [0,2,3,4,5,6,1,2,3,4,5,6,1];
17053 0 2
17054 0 2 begin
17055 0 2 with gle_gcb^ do
17056 0 2 begin
17057 0 3 info1 := ls_map[index]; { map DGL to GLE def }
17058 0 3 info2 := 4; { repeat rate 4% }
17059 0 3 info3 := 0; { linestyle mode }
17060 0 3 info4 := raster_patterns[index-1]; { pattern }
17061 0 3 gle_linestyle ( gle_gcb );
17062 0 3 end;
17063 0 2
17064 0 2
17065 0 2 end; { set_line_style }
17066 0 2
17067 0 1 function return_closest_color ( r,g,b : real; { target color }
17068 0 2 c_table_ptr : anyptr ) { system color map } : integer;
17069 0 2

```



```

17070:D -24 2 var
17071:D -24 2 target_h,
17072:D -24 2 target_s,
17073:D -48 2 target_l: real; { HSL target color values }
17074:D -48 2 error_h, { hue error distance squared }
17075:D -48 2 error_h2, { lightness error distance squared }
17076:D -48 2 error_l,
17077:D -48 2 error_s,
17078:D -48 2 error_l2,
17079:D -96 2 error_s2: real; { saturation error distance squared }
17080:D -96 2 map_h,
17081:D -96 2 map_s,
17082:D -96 2 map_l, { Current color map entry in HSL }
17083:D -128 2 error: real; { Distance from target to current color map entry }
17084:D -136 2 closest_error: real;
17085:D -136 2 i;
17086:D -144 2 closest_index: integer; { Best fit color map index }
17087:S
17088:D -144 2 { Find closest color from system color map, to match target color. }
17089:D -144 2 { The closest color is the color which 'looks' the closest. This }
17090:D -144 2 { algorithm has been derived from a mixture of logic and }
17091:D -144 2 { experimentation. The algorithm calculates for each entry in the }
17092:D -144 2 { color map an error factor indicating how far off the color map }
17093:D -144 2 { value is from the target color. It then returns the color map }
17094:D -144 2 { index with the least error. }
17095:S
17096:D -144 2 { Experimentation has showed that the best looking color is normally }
17097:D -144 2 { the color with the least error in hue. However when the target }
17098:D -144 2 { color is near black or white this is not true, and when many color }
17099:D -144 2 { map entries have a small hue error the closest hue does not produce }
17100:D -144 2 { the best color. The algorithm makes special cases out of the }
17101:D -144 2 { above cases and 'weights' the error result to reduce the effects }
17102:D -144 2 { of hue. }
17103:S
17104:C
17105:C 2 begin
17106:C 2 convert_rgb_to_hsl(r,g,b,target_h,target_s,target_l);
17107:C 2 closest_index := 1;
17108:C 2 closest_error := maxint; { worst case error }
17109:C 2 for i := 0 to gle_gcb^.gamut do { for each CMAP entry }
17110:C 2 with color_table_ptr_def(c_table_ptr)^[i] do { force anyptr to known type }
17111:C 2 begin
17112:C 2 convert_rgb_to_hsl(red,green,blue,map_h,map_s,map_l);
17113:C 4 { Calc errors, note that since Hue is circular it must }
17114:C 4 { be calc as shortest dist of either direction }
17115:S
17116:C 4 error_h := abs(map_h-target_h);
17117:C 4 error_h2 := abs(map_h-1-target_h);
17118:C 4 if error_h2 < error_h then error_h := error_h2;
17119:C 4 error_h2 := error_h * error_h;
17120:C 4 error_s := abs(map_s-target_s);
17121:C 4 error_l := abs(map_l-target_l);
17122:C 4 error_s2 := error_s * error_s;
17123:C 4 error_l2 := error_l * error_l;
17124:S
17125:C 4 if target_l < 0.1 then { special case where request is near black }
17126:C 4 begin
17127:C 4 { With small lum in cmap, sat and hue are undefined and can't }
17128:C 4 { be used in error calculation }
17129:C 4 if map_l < 0.01 then error := error_l

```

```

17130:C 6 else
17131:C 6 { With small sat in cmap, hue is undefined and can't be used }
17132:C 6 { in error calculation }
17133:C 6 if map_s < 0.01 then error := error_l + error_s2
17134:C 6 else error := error_l + error_h2 + error_s2;
17135:C 6 end
17136:C 6 else
17137:C 6 if target_s < 0.1 then { special case where request is near white }
17138:C 6 begin
17139:C 6 { With small lum in cmap, sat and hue are undefined and can't }
17140:C 6 { be used in error calculation }
17141:C 6 if map_l < 0.01 then error := 3
17142:C 6 else
17143:C 6 { With small sat in cmap, hue is undefined and can't be used }
17144:C 6 { in error calculation }
17145:C 6 if map_s < 0.01 then error := error_l2 + error_s
17146:C 6 else error := error_l2 + error_h2 + error_s;
17147:C 6 end
17148:C 6 else { normal case }
17149:C 6 { With small lum in cmap, sat and hue are undefined and can't }
17150:C 6 { be used in error calculation }
17151:C 6 if map_l < 0.01 then error := 3
17152:C 6 else
17153:C 6 { With small sat in cmap, hue is undefined and can't be used }
17154:C 6 { in error calculation }
17155:C 6 if map_s < 0.01 then error := 3
17156:C 6 else error := error_h2;
17157:C 6
17158:C 4 if error < closest_error then
17159:C 4 begin
17160:C 4 closest_error := error;
17161:C 4 closest_index := i;
17162:C 4 end;
17163:C 4 end;
17164:C 2 return_closest_color := closest_index;
17165:C 2 end;
17166:S
17167:S
17168:D 1 procedure raster_color ( index : integer );
17169:S
17170:D 2 var
17171:D -3 2 intensity: real;
17172:D -32 2 h,s,l: real;
17173:S
17174:C
17175:C 2 begin
17176:C 2 with gcb^,gle_gcb^ do
17177:C 2 begin
17178:C 2 with color_table_ptr^[index] do
17179:C 2 if gamut = 1 then { b&w }
17180:C 2 begin
17181:C 2 { Numbers from Dawn (4P-9000) DGL for 2648 terminal }
17182:C 2 intensity := 0.3*red + 0.59*green + 0.11*blue;
17183:C 2 if intensity < 0.06 then info1 := 0;
17184:C 2 else info1 := 1;
17185:C 2 end
17186:C 2 else
17187:C 2 if raster_device_rec_ptr(dev_dep_stuff)^devicetype = 2 then { moonunit }
17188:C 2 **WARNING: (line 17187): ADDR of a constant may not be supported on other implementations
17189:C 2 info1 := return_closest_color(red,green,blue,addr(info1_color_table))
17190:C 2 else

```



```

17189:C      6      if (index <= gamut) then info1 := index
17190:C      7      else
17191:C      8          info1 := return_closest_color(red,green,blue,color_table_ptr);
17192:C      9
17193:C     10      gle_index_color ( gle_gcb ); { this function sets color }
17194:C     11      end;
17195:C     12      end;
17196:C
17197:D     13      procedure raster_color_table ( index : integer;
17198:D     14          parm1 : real;
17199:D     15          parm2 : real;
17200:D     16          parm3 : real);
17201:C     17
17202:D     18      var
17203:D     19      color_list : array [1..3] of gle_shortint;
17204:D     20      intensity : real;
17205:C     21
17206:C     22      begin
17207:C     23      with gcb^,gle_gcb^ do
17208:C     24      begin
17209:C     25      if dgl_current_color_model = 2 then
17210:C     26      convert_hsl_to_rgb(parm1,parm2,parm3,parm1,parm2,parm3);
17211:C     27      if (index <= gamut) and (color_map_support = 1) then
17212:C     28      begin
17213:C     29      info1 := index;
17214:C     30      info2 := index;
17215:C     31      color_list[1] := trunc(parm1 * 1023 + 0.5);
17216:C     32      color_list[2] := trunc(parm2 * 1023 + 0.5);
17217:C     33      color_list[3] := trunc(parm3 * 1023 + 0.5);
17218:C     34      info_ptr1 := addr(color_list);
17219:C     35      gle_define_color_map ( gle_gcb );
17220:C     36      end;
17221:C     37      with color_table_ptr^[index] do
17222:C     38      begin
17223:C     39      red := parm1;
17224:C     40      green := parm2;
17225:C     41      blue := parm3;
17226:C     42      if index = 0 then
17227:C     43      begin
17228:C     44      if color_map_support = 1 then
17229:C     45      dgl_background_index := 0
17230:C     46      else
17231:C     47      begin
17232:C     48      if gamut = 1 then { b&w }
17233:C     49      begin
17234:C     50      { Numbers from Dawn (HP-9000) DGL for 2648 terminal }
17235:C     51      intensity := 0.3*red + 0.59*green + 0.11*blue;
17236:C     52      if intensity < 0.06 then dgl_background_index := 0
17237:C     53      else
17238:C     54      dgl_background_index := 1;
17239:C     55      end
17240:C     56      else { moonunit }
17241:C     57      dgl_background_index := return_closest_color(red,green,blue,addr(init_color_table)
17242:C     58      )
17243:C     59      end;
17244:C     60      end;
17245:C     61      end;
17246:C

```

```

17247:D     1      procedure set_all_color_table ( anyvar list : greal_list );
17248:C     2
17249:D     3      var
17250:D     4      color_list : array[0..47] of gle_shortint;
17251:D     5      parm1,
17252:D     6      parm2,
17253:D     7      parm3 : real;
17254:D     8      adr : gle_shortint;
17255:D     9
17256:C    10      begin
17257:C    11      with gcb^,gle_gcb^ do
17258:C    12      begin
17259:C    13      for i := 0 to 15 do
17260:C    14      begin
17261:C    15      adr := 1 * 3;
17262:C    16      parm1 := list[adr];
17263:C    17      parm2 := list[adr+1];
17264:C    18      parm3 := list[adr+2];
17265:C    19      if dgl_current_color_model = 2 then
17266:C    20      convert_hsl_to_rgb(parm1,parm2,parm3,parm1,parm2,parm3);
17267:C    21      color_list[adr] := trunc(parm1 * 1023 + 0.5);
17268:C    22      color_list[adr+1] := trunc(parm2 * 1023 + 0.5);
17269:C    23      color_list[adr+2] := trunc(parm3 * 1023 + 0.5);
17270:C    24      with color_table_ptr^[i] do
17271:C    25      begin
17272:C    26      red := parm1;
17273:C    27      green := parm2;
17274:C    28      blue := parm3;
17275:C    29      end;
17276:C    30      end;
17277:C    31      end;
17278:C
17279:C    32      info1 := 0;
17280:C    33      info2 := 15;
17281:C    34      info_ptr1 := addr(color_list);
17282:C    35      gle_define_color_map ( gle_gcb );
17283:C    36      end;
17284:C
17285:C
17286:D     1      procedure dummy_on_off ( gcb : graphics_control_block_ptr );
17287:C     2
17288:C     3      begin
17289:C     4      end;
17290:C
17291:D     1      procedure graphics_on_off ( gcb : graphics_control_block_ptr );
17292:C     2
17293:D     3      var
17294:D     4      on : boolean;
17295:C     5
17296:C     6      begin
17297:C     7      with gcb^ do
17298:C     8      begin
17299:C     9      on := info1 <> 0;
17300:C    10      if ( on and not graphicstate ) or
17301:C    11      ( not on and graphicstate ) then
17302:C    12      call (togglegraphicshook);
17303:C    13      end;
17304:C    14      end;
17305:C
17306:D     1      procedure dump_graphics ( mask : integer );

```



```

17307:S
17308:D 2 { Purpose: To dump bit map to standard printer }
17309:S
17310:D 2 label 1;
17311:S
17312:D 2 const
17313:D 2 gbufferSize = 255;
17314:S
17315:D 2 var
17316:D -256 2 gbuffer : packed array [1..gbufferSize] of char;
17317:D -260 2 y : integer;
17318:D -264 2 bytes_wide : integer;
17319:S
17320:D
17321:C 2 begin
17322:C 2 gbuffer[1] := chr(27); { escape sequence for graphics }
17323:C 2 gbuffer[2] := 'x';
17324:C 2 gbuffer[3] := 'b';
17325:C 2 gbuffer[4] := '6';
17326:C 2 gbuffer[5] := '4';
17327:C 2 gbuffer[6] := 'W';
17328:S
17329:C 2 with gle_gcb do
17330:C 3 begin
17331:C 3 bytes_wide := (display_max_x - display_min_x + 8) div 8;
17332:C 3 info_ptr1 := addr(gbuffer[7]);
17333:C 3 info1 := mask;
17334:S
17335:C 3 for y := display_min_y to display_max_y do
17336:C 4 begin
17337:C 4 info2 := y;
17338:C 4 gle_get_raster ( gle_gcb );
17339:C 4 write(gfiles[4]^,gbuffer:bytes_wide+6);
17340:C 4 if ioresult <> ord(inoerror) then goto 1;
17341:C 4 end;
17342:C 3 end;
17343:C 2 write(gfiles[4]^,#27*'rB'); { terminate graphics sequence }
17344:D 1;
17345:C 2 end;
17346:S
17347:D 1 procedure raster_input_esc ( opcode : integer;
17348:D 2 isize : integer;
17349:D 2 rsize : integer;
17350:D 2 anyvar ilist : gint_list;
17351:D 2 anyvar rlist : greal_list;
17352:D 2 var ierr : integer );
17353:D
17354:S
17355:D 2 { Purpose : To perform an input escape function }
17356:S
17357:D 2 begin
17358:D 2 ierr := 1; { no input escape functions supported }
17359:S
17360:D 2 end; { input_esc }
17361:S
17362:S
17363:D 1 {***** gator dumper *****}
17364:D 1 procedure dumpgator ;
17365:D 2 label 1;
17366:S

```

```

17367:D 2 const
17368:D 2 gwidthb = 128;
17369:D 2 gmaxheight = 612;
17370:D 2 gbufferSize = gwidthb + 7;
17371:S
17372:D 2 type
17373:D 2 gbyte = 0..255;
17374:D 2 row_def = packed array [0..(1024*768)-1] of gbyte;
17375:S
17376:D 2 var
17377:D -4 2 row : ^row_def;
17378:S
17379:D -140 2 gbuffer : packed array [1..gbufferSize] of char;
17380:D -148 2 i,j : integer;
17381:D -152 2 index : integer;
17382:D -156 2 bit_mask : integer;
17383:D -160 2 result : integer;
17384:D -164 2 tmp : ^horint;
17385:D -168 2 tmp2 : integer;
17386:S
17387:C 2 begin
17388:S
17389:C 2 tmp := anyptr(control_space + 16384);
17390:C 2 tmp2 := ((tmp) mod 16) * 1048576;
17391:C 2 row := anyptr(tmp2);
17392:S
17393:C 2 write(gfiles[4]^,#27*'rA'); { initiate graphics sequence }
17394:S
17395:C 2 gbuffer[1] := chr(27); { escape sequence for graphics }
17396:C 2 gbuffer[2] := 'x';
17397:C 2 gbuffer[3] := 'b';
17398:C 2 gbuffer[4] := '1';
17399:C 2 gbuffer[5] := '2';
17400:C 2 gbuffer[6] := '8';
17401:C 2 gbuffer[7] := 'W';
17402:S
17403:C 2 for j := 0 to 767 do
17404:C 3 begin
17405:C 3 for i := 0 to 127 do
17406:C 4 begin
17407:C 4 result := 0;
17408:C 4 index := j*1024+i*8;
17409:C 4 bit_mask := 256;
17410:C 4 for index := index to index+7 do
17411:C 5 begin
17412:C 5 bit_mask := bit_mask div 2;
17413:C 5 if odd(row^[index]) then result := bit_mask+result;
17414:C 5 end;
17415:C 4 gbuffer[i+8] := chr(result);
17416:C 4 end;
17417:C 3 write(gfiles[4]^,gbuffer:gwidthb+7);
17418:C 3 if ioresult <> ord(inoerror) then goto 1;
17419:C 3 end;
17420:S
17421:C 2 write(gfiles[4]^,#27*'rB'); { terminate graphics sequence }
17422:C 2 1;
17423:C 2 end; { of crt }
17424:D 1 {***** end gator dumper *****}
17425:S
17426:D 1 procedure raster_output_esc ( opcode : integer;

```



```

17421:D      4      isize : integer;
17422:D      4      rsize : integer;
17423:D      4      anyvar ilist : gint_list;
17430:D      2      anyvar rlist : greal_list;
17432:D      2      var ierr : integer );
17433:D      2 ( Purpose : To perform an output escape function )
17434:S
17435:D      2 var
17436:D      -1 2 on : boolean;
17437:S
17438:S
17439:S
17440:C      2 begin
17441:C      2 with gle_gcb^, raster_device_rec_ptr(dev_dep_stuff)^ do
17442:C      3 if (opcode = 52) and (devicetype <> 4) and (devicetype <> 2) then
17443:C      4 begin
17444:C      4 if ierr = 0 then call (dumpgraphicshook);
17445:C      4 end
17446:C      4 else
17447:C      4 if (opcode = 52) and (devicetype = 2) then
17448:C      5 begin
17449:C      5 if ierr = 0 then dump_graphics(-1);
17450:C      5 end
17451:C      5 else
17452:C      5 if (opcode = 52) and (devicetype = 4) then {added for gator case}
17453:C      6 begin
17454:C      6 if ierr = 0 then dumpgat;
17455:C      6 end
17456:C      6 else
17457:C      6 if (opcode = 53) and (devicetype = 3) then
17458:C      7 begin
17459:C      7 if ierr = 0 then gle_await_blanking ( gle_gcb );
17460:C      7 end
17461:C      7 else
17462:C      7 if (opcode = 250) and
17463:C      8 ((devicetype = 0) or (devicetype = 2)) then
17464:C      8 begin { marmot and aspen and moonunit}
17465:C      8 if ierr = 0 then
17466:C      9 begin
17467:C      9 if (rlist[1] > 0.0) and ( rlist[2] > 0.0 ) then
17468:C      10 begin
17469:C      10 display_res_x := rlist[1];
17470:C      10 display_res_y := rlist[2];
17471:C      10 end
17472:C      10 else
17473:C      10 ierr := 4;
17474:C      9 end;
17475:C      8 end;
17476:C      8 else
17477:C      8 if (opcode = 1050) and (devicetype <> 4) then { graphics on / off }
17478:C      9 begin
17479:C      9 if ierr = 0 then
17480:C      10 begin
17481:C      10 info1 := ilist[1];
17482:C      10 gle_graphics_on_off ( gle_gcb );
17483:C      10 end;
17484:C      9 end
17485:C      9 else
17486:C      9 if (opcode = 1051) and ((devicetype <> 2) or (devicetype <> 4))

```

```

17487:C      10 then { alpha on / off }
17488:C      10 begin
17489:C      10 if ierr = 0 then
17490:C      11 begin
17491:C      11 on := ilist[1] <> 0;
17492:C      11 if ( on and not alphastate ) or
17493:C      12 ( not on and alphastate ) then
17494:C      12 call(togglealphahook);
17495:C      11 end;
17496:C      10 end
17497:C      10 else
17498:C      10 if (opcode = 1052) then
17499:C      11 begin
17500:C      11 if ierr = 0 then
17501:C      12 begin
17502:C      12 info1 := ilist[1];
17503:C      12 if (info1 < 0) or (info1 > 3) then info1 := 0;
17504:C      12 if info1 = 1 then info1 := 2
17505:C      13 else
17506:C      13 if info1 = 2 then info1 := 1;
17507:C      12 gle_define_drawing_mode ( gle_gcb );
17508:C      12 end;
17509:C      11 end
17510:C      11 else
17511:C      11 if (opcode = 1053) and
17512:C      12 ((devicetype = 3) or (devicetype = 2)) then
17513:C      12 begin
17514:C      12 if ierr = 0 then dump_graphics(ilist[1]);
17515:C      12 end
17516:C      12 else
17517:C      12 if (opcode = 1053) and (devicetype = 4) then {for gator}
17518:C      13 begin
17519:C      13 if ierr = 0 then dumpgat;
17520:C      13 end
17521:C      13 else
17522:C      13 if (opcode = 1054) and (devicetype <> 4) then
17523:C      14 begin
17524:C      14 if ierr = 0 then
17525:C      15 begin
17526:C      15 info1 := ilist[1];
17527:C      15 info2 := 0;
17528:C      15 gle_clear(gle_gcb);
17529:C      15 end;
17530:C      14 end
17531:C      14 else
17532:C      14 if (opcode = 10050) and (devicetype = 3) then
17533:C      15 begin
17534:C      15 if (ierr = 3) and (rsize = 48) { opcode_ck gave real size error } then
17535:C      16 begin
17536:C      16 ierr := 0;
17537:C      16 set_all_color_table(rlist);
17538:C      16 end;
17539:C      15 end
17540:C      15 else
17541:C      15 ierr := 1;
17542:C      15 end; { raster_output_esc }
17543:S
17544:D      1 procedure dgl_raster_init ( control : integer );
17545:S
17546:D      2 type

```



```

17547:D      2  default_poly_table_def = array[1..16] of poly_entry_def;
17548:S
17549:D      2  control_def = packed record
17550:D          case gshortint of
17551:D              0 : (whole : gshortint);
17552:D              1 : (part : packed record
17553:D                  b15,b14,b13,b12,
17554:D                  b11,b10,b9, b8,
17555:D                  clr_inhibit,b6,b5,b4,
17556:D                  b3,b2,b1,b0 : boolean;
17557:D              end);
17558:D          end;
17559:S
17560:D      2  const
17561:D          default_poly_table = default_poly_table_def [
17562:D              poly_entry_def [ density : 0.0, orient : 0.0, edge : true ], ( 1 )
17563:D              poly_entry_def [ density : 0.125, orient : 90.0, edge : true ], ( 2 )
17564:D              poly_entry_def [ density : 0.125, orient : 0.0, edge : true ], ( 3 )
17565:D              poly_entry_def [ density : -0.125, orient : 0.0, edge : true ], ( 4 )
17566:D              poly_entry_def [ density : 0.125, orient : 45.0, edge : true ], ( 5 )
17567:D              poly_entry_def [ density : 0.125, orient : -45.0, edge : true ], ( 6 )
17568:D              poly_entry_def [ density : -0.125, orient : 45.0, edge : true ], ( 7 )
17569:D              poly_entry_def [ density : 0.25, orient : 90.0, edge : true ], ( 8 )
17570:D              poly_entry_def [ density : 0.25, orient : 0.0, edge : true ], ( 9 )
17571:D              poly_entry_def [ density : -0.25, orient : 0.0, edge : true ], ( 10 )
17572:D              poly_entry_def [ density : 0.25, orient : 45.0, edge : true ], ( 11 )
17573:D              poly_entry_def [ density : 0.25, orient : -45.0, edge : true ], ( 12 )
17574:D              poly_entry_def [ density : -0.25, orient : 45.0, edge : true ], ( 13 )
17575:D              poly_entry_def [ density : -0.5, orient : 0.0, edge : true ], ( 14 )
17576:D              poly_entry_def [ density : 1.0, orient : 0.0, edge : false ], ( 15 )
17577:D              poly_entry_def [ density : 1.0, orient : 0.0, edge : true ], ( 16 )
17578:S
17579:D      2  var
17580:D          temp_control : control_def;
17581:D          i : integer;
17582:D      -10  temp_color_model : integer;
17583:D      -18  c : real;
17584:S
17585:C      2  begin
17586:C          if gle_gcb^.display_name = '9837a' then
17587:C              gle_gcb^.graphics_on_off := dummy_on_off
17588:C          else
17589:C              if gle_gcb^.display_name <> '98627a' then
17590:C                  gle_gcb^.graphics_on_off := graphics_on_off;
17591:C              with gle_gcb^.gcb^.raster_device_rec_ptr(dev_dep_stuff)^ do
17592:C                  begin
17593:C                      disp_just := centered;
17594:C                      clipping_support := true;
17595:C                      retroactive_polygon_support := false;
17596:C                      retroactive_color_support := color_map_support = 1;
17597:C                      number_markers := 19;
17598:C                      number_dgl_linestyles := 8;
17599:C                      maximum_polygon_vertices := 32767;
17600:C                  end;
17601:C          3  proc_output_esc := raster_output_esc;
17602:C          3  proc_input_esc := raster_input_esc;
17603:C          3  proc_linestyle := raster_linestyle;
17604:C          3  proc_color := raster_color;
17605:C          3  proc_color_table := raster_color_table;
17606:C

```

```

17607:S      3  temp_control.whole := control;
17608:C      3  if not temp_control.part.clr_inhibit then
17609:C          with gle_gcb^ do
17610:C              begin
17611:C                  info1 := -1;           ( clear all planes )
17612:C                  info2 := 0;
17613:C                  gle_clear ( gle_gcb );
17614:C              end;
17615:C          end;
17616:S      3  ( allocate color table space )
17617:C      3  temp_color_model := dgl_current_color_model;
17618:S      3  dgl_current_color_model := 1; ( rgb )
17619:C      3  color_table_size := 31;
17620:C      3  color_table_ptr := addr(color_table_def_space);
17621:S      3  if (gamut > 1) then
17622:C          begin
17623:C              for i := 0 to 15 do
17624:C                  with init_color_table[i] do
17625:C                      raster_color_table(i,red,green,blue);
17626:C                  raster_color_table(i6,i,i,i);
17627:C              end
17628:C          else
17629:C              begin
17630:C                  raster_color_table(0,0,0,0);
17631:C                  for i := 1 to 16 do
17632:C                      begin
17633:C                          c := ((17-i) / 16);
17634:C                          raster_color_table(i,c,c,c);
17635:C                      end;
17636:C                  end;
17637:C          end;
17638:C      3  for i := 17 to 31 do
17639:C          raster_color_table(i,1,1,1);
17640:C      3  dgl_current_color_model := temp_color_model;
17641:S      3  gle_gcb^.info1 := 1;
17642:C      3  gle_index_color( gle_gcb );
17643:S      3  ( allocate polygon table space )
17644:C      3  number_polygon_styles := poly_table_size;
17645:C      3  poly_table_ptr := addr(poly_table_def_space);
17646:C      3  for i := 1 to poly_table_size do
17647:C          poly_table_ptr[i] := default_poly_table[i];
17648:S      3  display_echo_mult := 1;
17649:C      3  end;
17650:C      2  end;
17651:S      1  end. ( dgl_raster )
17652:S

```

No errors. 2 warnings.

**** Nonstandard language features enabled ****

DGL_TOOLS

Description

DGL_TOOLS provides the function RETURN_MACHINE_TYPE.

Requirements

GLE_TYPES.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:0 { Graphics Low End
2:0 {
3:0 {
4:0 { Module = DGL_TOOLS
5:0 { Programmer = BJS
6:0 { Date = 12-07-82
7:0 {
8:0 { Purpose: To provide general tools for GLE and DGL.
9:0 {
10:0 { Rev history
11:0 { Created - 12-07-82
12:0 { Modified - 9-07-83 BJS Changed to add check for marbox
13:0 {
14:0 { (c) Copyright Hewlett-Packard Company, 1983.
15:0 { All rights are reserved. Copying or other
16:0 { reproduction of this program except for archival
17:0 { purposes is prohibited without the prior
18:0 { written consent of Hewlett-Packard Company.
19:0 {
20:0 {
21:0 {
22:0 {
23:0 {
24:0 {
25:0 {
26:0 {
27:0 {
28:0 {
29:0 {
30:0 {
31:0 {
32:0 {
33:0 {
34:0 {
35:0 {
36:0 {
37:0 {
38:0 {
39:0 {
40:0 {
41:0 {
42:0 {
43:0 {
44:0 {
45:0 {
2000:0 {
2001:0 {
2002:0 {
2003:0 {
2004:0 {
2005:0 {
2006:0 {
2007:0 {
2008:0 {
2009:0 {
2010:0 {
2011:0 {
2012:0 {
2013:0 {

```

```

20014:S 2 type
20015:D 2 sysflag_def = packed record
20016:D 2 bit7,bit6,hpib,crt_config,
20017:D 2 kbd,high,big_graph,alpha50 : boolean;
20018:D 2 end;
20019:D 2
20020:S 2
20021:D 2 crt_reg_def = packed record
20022:D 2 selfinit,bit14,bit13,top1,top2,highlite,
20023:D 2 graph,alpha,
20024:D 2 bit7,bit6,bit5,bit4,bit3,bit2,bit1,bit0 : boolean;
20025:D 2 end;
20026:S 2
20027:D 2 var
20028:D 2 sysflag [ hex('ffffd2') ] : sysflag_def;
20029:D 2 crt_reg [ hex('S1ffff') ] : crt_reg_def;
20030:S 2
20031:C 2 begin
20032:C 2 if sysflag.big_graph then
20033:C 2 begin { could be 9836A or 9836C or Marbox }
20034:C 2 if sysflag.crt_config then { might be 9836C or Marbox }
20035:C 2 begin
20036:C 2 CRT reg bits 12 (top1) and 11 (top2) are defined
20037:C 2 as follows:
20038:C 2 {
20039:C 2 00 - Monochrome
20040:C 2 01 - 4 planes starting at $520000
20041:C 2 10 - 1 pixel / byte; 4 bits / pixel
20042:C 2 11 - 3 planes starting at $528000
20043:C 2 8 pixel / byte (moonunit like)
20044:C 2 11 - 8 planes starting at $520000
20045:C 2 1 pixel / byte; 8 bits / pixel
20046:C 2 }
20047:C 2 with crt_reg do
20048:C 2 if (not top1) and (top2) then
20049:C 2 return_machine_type := m9836c { 9836C }
20050:C 2 else
20051:C 2 if (not top1) and (not top2) then
20052:C 2 return_machine_type := m9836a { Marbox }
20053:C 2 else
20054:C 2 return_machine_type := munknown; { unknown }
20055:C 2 end
20056:C 2 else
20057:C 2 begin
20058:C 2 return_machine_type := m9836a; { 9836A }
20059:C 2 end;
20060:C 2 end
20061:C 2 else
20062:C 2 if sysflag.alpha50 then
20063:C 2 begin
20064:C 2 return_machine_type := m9826a; { 9826A }
20065:C 2 end
20066:C 2 else
20067:C 2 begin
20068:C 2 return_machine_type := m9816a; { 9816A }
20069:C 2 end;
20070:C 2 end;
20071:S 1 end

```


DGL_VARS

Description

DGL_VARS contains most of the graphics library global variables and DGL types.

Requirements

DGL_TYPES and GLE_TYPES.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:0      0 {
2:0      0 { Pascal work station graphics library
3:0      0 {
4:0      0 { Module = DGL_VARS
5:0      0 { Programmer = EJS
6:0      0 { Date = 2/1/82
7:0      0 {
8:0      0 { Purpose: To hold graphics library global variables
9:0      0 {
10:0     0 { Rev history
11:0     0 { 5-13-82 BJS - Made changes to add hp9816 support
12:0     0 { 5-24-82 BJS - Made changes to support inq of world cp values
13:0     0 { 7-01-82 BJS - Made changes to support 8 color HP 9836C
14:0     0 { 8-25-82 BJS - Major mods for addition of GLE
15:0     0 { 2-17-84 BDS - Changed gcb vars to global for Pascal 3.0
16:0     0 {
17:0     0 { (c) Copyright Hewlett-Packard Company, 1983.
18:0     0 { All rights are reserved. Copying or other
19:0     0 { reproduction of this program except for archival
20:0     0 { purposes is prohibited without the prior
21:0     0 { written consent of Hewlett-Packard Company.
22:0     0 {
23:0     0 {
24:0     0 { RESTRICTED RIGHTS LEGEND
25:0     0 {
26:0     0 { Use, duplication, or disclosure by the Government
27:0     0 { is subject to restrictions as set forth in
28:0     0 { paragraph (b) (3) (B) of the Rights in Technical
29:0     0 { Data and Computer Software clause in
30:0     0 { DAR 7-104.9(a).
31:0     0 {
32:0     0 { HEWLETT-PACKARD COMPANY
33:0     0 { Fort Collins, Colorado
34:0     0 {
35:0     0 { $modcal$,
36:0     0 { $search 'TYPES',
37:0     0 { $include 'GLE_LIB'$
38:0     0 { $include 'OPTIONS'$
39:0     0 { { This include file specifies range checking, debug and other compiler
40:0     0 { options for the graphics library
41:0     0 {
42:0     0 { $debug OFF$
43:0     0 { $range OFF$
44:0     0 { $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
45:0     0 { $FLOAT_AOW TEST$
46:0     0 {
47:0     0 {
48:0     0 {
49:0     0 {
50:0     0 { $include 'OPTIONS'$
51:0     0 {
52:0     0 { module DGL_VARS;
53:0     0 {
54:0     1 import DGL_TYPES,GLE_TYPES;
55:0     0 {
56:0     1 export
57:0     0 {
58:0     1 $linenum 1000$
59:0     0 {
60:0     0 {

```

```

1000:0   1 const
1001:0   1   graphics_rev = '3.0'; { revision number }
1002:0   1   default_color_table_size = 32;
1003:0   1   default_poly_table_size = 16;
1004:0   1   poly_table_size = 16;
1005:0   1
1006:0   1
1007:0   1
1008:0   1
1009:0   1 type
1010:0   1   bounds = record
1011:0   1     xmin : real;
1012:0   1     xmax : real;
1013:0   1     ymin : real;
1014:0   1     ymax : real;
1015:0   1   end;
1016:0   1
1017:0   1   vir_bounds = record
1018:0   1     xlim : real;
1019:0   1     ylim : real;
1020:0   1   end;
1021:0   1
1022:0   1 type
1023:0   1   c_def = packed record
1024:0   1     red : real;
1025:0   1     green : real;
1026:0   1     blue : real; { 24 bytes each }
1027:0   1   end;
1028:0   1
1029:0   1   color_table_def = array [0..31] of c_def;
1030:0   1   color_table_ptr_def = ^color_table_def;
1031:0   1
1032:0   1   poly_entry_def = packed record
1033:0   1     density : real;
1034:0   1     orient : real;
1035:0   1     edge : boolean; { 18 bytes each }
1036:0   1   end;
1037:0   1
1038:0   1   poly_table_def = array [1..poly_table_size] of poly_entry_def;
1039:0   1   poly_table_ptr_def = ^poly_table_def;
1040:0   1
1041:0   1 { a large number of system variables are allocated in dynamic memory
1042:0   1 { to conserve the amount of global space used. These variables
1043:0   1 { are all part of the GCB (graphics control block)
1044:0   1 {
1045:0   1   graphics_control_block1 =
1046:0   1   record
1047:0   1     aspect_ratio : real; { current aspect ratio }
1048:0   1
1049:0   1     log_aspect : real; { aspect ratio of the logical display limits }
1050:0   1
1051:0   1     log_disp_lim : bounds; { logical display limits }
1052:0   1
1053:0   1     max_disp_lim : bounds; { maximum display limits }
1054:0   1
1055:0   1     def_disp_lim : bounds; { default display limits }
1056:0   1
1057:0   1     log_loc_lim : bounds; { logical locator limits }
1058:0   1
1059:0   1     max_loc_lim : bounds; { maximum locator limits }
1060:0   1
1061:0   1     def_loc_lim : bounds; { default locator limits }

```



```

1062:S
1063:D 1 window_lim : bounds; { current window }
1064:S
1065:D 1 viewport_lim : bounds; { current viewport }
1066:S
1067:D 1 cur_disp_lim : bounds; { current display coordinate limits }
1068:S
1069:D 1 cur_vir_lim : vir_bounds; { current virtual limits }
1070:S
1071:D 1 dgl_char_width : real; { current character width (world) }
1072:S
1073:D 1 dgl_char_height : real; { current character height (world) }
1074:S
1075:D 1 char_rot_h : real; { char rot cos vector }
1076:S
1077:D 1 char_rot_w : real; { char rot sin vector }
1078:S
1079:D 1 disp_just : (centered,lowerleft); { display justification }
1080:S
1081:D 1 dxunits : real; { # of units in the logical display coord. }
1082:D 1 dyunits : real; { system. }
1083:S
1084:D 1 number_dgl_linestyles : gshortint;
1085:D 1 number_markers : gshortint;
1086:D 1 number_polygon_styles : gshortint;
1087:D 1 color_table_size : gshortint;
1088:S
1089:D 1 dgl_current_color : gshortint;
1090:D 1 dgl_current_linestyle : gshortint;
1091:D 1 dgl_current_linewidth : gshortint;
1092:D 1 dgl_current_timing_mode : gshortint;
1093:D 1 dgl_current_polygon_color : gshortint;
1094:D 1 dgl_current_polygon_style : gshortint;
1095:D 1 dgl_current_polygon_linestyle : gshortint;
1096:D 1 dgl_current_polygon_angle : real;
1097:D 1 dgl_current_polygon_density : real;
1098:D 1 dgl_current_polygon_edge : boolean;
1099:D 1 dgl_current_polygon_crosshatch : boolean;
1100:D 1 dgl_current_color_model : gshortint;
1101:S
1102:D 1 maximum_polygon_vertices : gshortint;
1103:S
1104:D 1 retroactive_polygon_support : boolean;
1105:D 1 retroactive_color_support : boolean;
1106:S
1107:D 1 clipping_support : boolean;
1108:S
1109:D 1 disp_dev_adr : integer; { display device address }
1110:D 1 disp_file_name : gstring2SS; { name of device file, knull if
1111:D 1 { device address is used }
1112:D 1 loc_dev_adr : integer; { locator device address }
1113:S
1114:D 1 disp_eq_loc : boolean; { true if loc and disp are same device }
1115:S
1116:D 1 poly_table_ptr : poly_table_ptr_def;
1117:S
1118:D 1 color_table_ptr : color_table_ptr_def;
1119:S
1120:D 1 marker_size_x : integer; { marker size in device units }
1121:D 1 marker_size_y : integer;

```

```

1122:S
1123:D 1 proc_output_esc : procedure ( opc, isize, rsize : integer;
1124:D 1 anyvar ilist : gint_list;
1125:D 1 anyvar rlist : greal_list;
1126:D 1 var error : integer);
1127:S
1128:D 1 proc_input_esc : procedure ( opc, isize, rsize : integer;
1129:D 1 anyvar ilist : gint_list;
1130:D 1 anyvar rlist : greal_list;
1131:D 1 var error : integer);
1132:S
1133:D 1 proc_linestyle : procedure ( index : integer );
1134:S
1135:D 1 proc_color : procedure ( index : integer );
1136:D 1
1137:D 1 proc_color_table : procedure ( index : integer ; p1,p2,p3 : real );
1138:S
1139:D 1 dgl_background_index : integer;
1140:S
1141:D 1 proc_await_locator : procedure ( var echo : integer; var button : integer;
1142:D 1 var x,y : real );
1143:S
1144:D 1 proc_sample_locator : procedure ( echo : integer; var x,y : real );
1145:S
1146:D 1 dgl_polygon_color_current : boolean; { true if polygon color set in gle }
1147:D 1 end;
1148:S
1149:D 1 const
1150:D eight_diget_epsilon = D.0000D0D1; { a number which will change the
1151:D value of a 8 diget number when
1152:D added to it }
1153:S
1154:D 1 { Initialization constants follow }
1155:S
1156:D 1 init_color = 1;
1157:D 1 init_linestyle = 1;
1158:D 1 init_linewidth = 1;
1159:D 1 init_char_width = 0.07;
1160:D 1 init_char_height = 0.1;
1161:D 1 init_char_width_factor = 0.035;
1162:D 1 init_char_height_factor = 0.05;
1163:D 1 init_char_rot_w = 1.0;
1164:D 1 init_char_rot_h = 0.0;
1165:S
1166:D 1 init_timing_mode = 0; { This is diff from HP 9000 due to PuS 1.0 compatibility }
1167:D 1 init_cpx = 0;
1168:D 1 init_cpy = 0;
1169:S
1170:D 1 init_dev_adr = 0;
1171:S
1172:D 1 init_window = bounds [ xmin : -1.0,
1173:D 1 xmax : 1.0,
1174:D 1 ymin : -1.0,
1175:D 1 ymax : 1.0];
1176:S
1177:D 1 init_viewport = bounds [ xmin : 0.0,
1178:D 1 xmax : 1.0,
1179:D 1 ymin : 0.0,
1180:D 1 ymax : 1.0];
1181:S

```



```

1182 D 1 init_aspect = 1.0;
1183 S
1184 D 1 init_vir_lim = vir_bounds [ xlim : 1.0,
1185 D 1 ylim : 1.0];
1186 S
1187 D 1 init_display_lim = bounds [ xmin : 0.0,
1188 D 1 xmax : 800000.0,
1189 D 1 ymin : 0.0,
1190 D 1 ymax : 800000.0];
1191 S
1192 D 1 init_locator_lim = bounds [ xmin : 0.0,
1193 D 1 xmax : 800000.0,
1194 D 1 ymin : 0.0,
1195 D 1 ymax : 800000.0];
1196 S
1197 S
1198 D 1 { definitions of some standard colors (used only on raster displays) }
1199 D 1 { that are used in locator echoing }
1200 S
1201 D 1 dominate_mode = 0;
1202 D 1 non_dominant_mode = 1;
1203 D 1 erase_mode = 2;
1204 D 1 complement_mode = 3;
1205 S
1206 D 1 solid_linestyle = 1;
1207 S
1208 D 1 io_error_number = -26;
1209 D 1 graphics_error_number = -27;
1210 S
1211 D 1 { definitions of some std device adrs }
1212 S
1213 D 1 internal_display = 3;
1214 D 1 internal_locator = 2;
1215 S
1216 D 1 { definitions of some defaults used for raster linestyle generation }
1217 S
1218 D 1 initial_pattern = -1;
1219 S
1220 D 1 { The following const are for user errors }
1221 S
1222 D 1 err_sys_int = 1; { system not initialized }
1223 D 1 err_dis_int = 2; { display not initialized }
1224 D 1 err_loc_int = 3; { locator not initialized }
1225 D 1 err_echo_dis_int = 4; { echo needs display initialized }
1226 D 1 err_aspect = 6; { illegal aspect ratio }
1227 D 1 err_bad_parms = 7; { illegal parameters }
1228 D 1 err_out_phys = 8; { parms outside physical disp lim }
1229 D 1 err_out_wind = 9; { parms outside window lim }
1230 D 1 err_disp_eq_loc = 10; { loc lim given when disp and loc are the same }
1231 D 1 err_out_virt = 11; { parms outside virt lim }
1232 D 1 err_no_display_hardware = 12; { missing display hardware }
1233 D 1 err_out_loc = 13; { parms outside loc lim }
1234 D 1 err_no_table = 14; { device does not support color table }
1235 D 1 err_neg_points = 16; { polygon npts < 0 }
1236 S
1237 D 1 var
1238 D 1 { define system initialization variables }
1239 S
1240 D -1 1 system_init : boolean; { is the system init }
1241 D -2 1 disp_init : boolean; { is the display init }

```

```

1242 D -3 1 loc_init : boolean; { is the locator init }
1243 S
1244 D -8 1 graphics_error : integer; { holds last error number }
1245 S
1246 D -12 1 gcb : ^graphics_control_block; { pointer to the dynamic vars }
1247 S
1248 D -12 1 { define holders of the current position }
1249 S
1250 D -16 1 cpy : integer;
1251 D -20 1 cpx : integer;
1252 D -20 1 world_int_cpx : gshortint; { last int_move / int_line cp value }
1253 D -24 1 world_int_cpy : gshortint;
1254 S
1255 S
1256 D -25 1 int_cp : boolean; { true if last move or line was integer }
1257 S
1258 D -34 1 world_real_cpx : real; { last move / line cp value }
1259 D -42 1 world_real_cpy : real;
1260 S
1261 D -42 1 { define the holders of the locator echo position }
1262 D -42 1 { Note that the echo is set by two routines (set_echo_pos,
1263 D -42 1 { calculate viewing) and both a world cord value and a device cord
1264 D -42 1 { value is kept. This was done since the device dependent value is
1265 D -42 1 { needed as a global value during echoing, yet the world cord value
1266 D -42 1 { is needed as a return value for some 'snap' echoes }
1267 S
1268 D -50 1 w_loc_echo_x : real; { world units }
1269 D -58 1 w_loc_echo_y : real;
1270 S
1271 D -62 1 d_loc_echo_x : integer; { device units }
1272 D -66 1 d_loc_echo_y : integer;
1273 S
1274 D -66 1 { conversion factors }
1275 S
1276 D -74 1 xwtod_scale : real; { world to display scale }
1277 D -82 1 ywtod_scale : real;
1278 S
1279 D -90 1 xdtow_scale : real; { display to world scale }
1280 D -98 1 ydtow_scale : real;
1281 S
1282 D -106 1 xwtod_offset : real; { world to display translation }
1283 D -114 1 ywtod_offset : real;
1284 S
1285 D -122 1 xdtow_offset : real; { display to world translation }
1286 D -130 1 ydtow_offset : real;
1287 S
1288 D -138 1 xltod_scale : real; { locator to display scale }
1289 D -146 1 yltod_scale : real;
1290 S
1291 D -147 1 calc_text_xform : boolean; { true if text xform needs to be recalcd }
1292 S
1293 D -147 1 scalef : record { used for int_move / int_line }
1294 D -147 1 x_display_delta : gshortint;
1295 D -147 1 x_window_delta : gshortint;
1296 D -147 1 x_display_offset : gshortint;
1297 D -147 1 y_display_delta : gshortint;
1298 D -147 1 y_window_delta : gshortint;
1299 D -147 1 y_display_offset : gshortint;
1300 D -160 1 end;
1301 S

```



```

1302:D -160 1 type
1303:D -160 1 ls_patterns = packed array [0..7] of gshortint;
1304:S
1305:D -160 1 const
1306:D -160 1 raster_patterns =
1307:D -160 1 ls_patterns [ -1 ($FFFF) { ..... }, { 1 }
1308:D -160 1 -256 ($FF00) { ..... }, { 2 }
1309:D -160 1 -64 ($FFC0) { ..... }, { 3 }
1310:D -160 1 -6 ($FFFA) { ..... }, { 4 }
1311:D -160 1 -10 ($FFF6) { ..... }, { 5 }
1312:D -160 1 -220 ($FF24) { ..... }, { 6 }
1313:D -160 1 -32640 ($8080) { ..... }, { 7 }
1314:D -160 1 -21846 ($AAAA) { ..... }, { 8 }
1315:S
1316:D -160 1 var
1317:D -161 1 short_flag : boolean; { true if int_move / int_line }
1318:D -161 1 { can used fast internal routines }
1319:S
1320:D -162 1 short_defaults : boolean; { true if viewport bounds }
1321:D -162 1 { map to max edges of raster display }
1322:S
1323:D -162 1 { input info }
1324:S
1325:D -166 1 current_echo_type : integer;
1326:D -170 1 display_echo_mult : integer;
1327:S
1328:D -174 1 cursor_size_x : integer;
1329:D -178 1 cursor_size_y : integer;
1330:S
1331:D -182 1 cursor_color : integer;
1332:S
1333:D -186 1 gle_gcb : graphics_control_block_ptr;
1334:D -190 1 gle_gcb1 : graphics_input_control_block_ptr;
1335:S
1336:D -194 1 gle_knob_echo_gcb : graphics_control_block_ptr; { for echo on internal
1337:D -194 1 crt even if display is }
1338:D -194 1 moonunit
1339:D -194 1 gcb_space : graphics_control_block1;
1340:D -194 1 gle_gcb_space : graphics_control_block;
1341:D -194 1 gle_gcb1_space : graphics_input_control_block;
1342:D -194 1 gle_knob_echo_gcb_space : graphics_control_block;
1343:D -194 1 color_table_def_space : color_table_def;
1344:D -194 1 poly_table_def_space : poly_table_def;
1345:D -3672 1 const
1346:D -3672 1 dno = 0;
1347:D -3672 1 dyes = 1;
1348:S
1349:D -3672 1 implement
1350:S
1351:C 1 end. { of module DGL_VAR }
1352:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

Description

DI_DRV provides low-level driver support.

Usage

DI_DRV is used with the 98625 interface.

Requirements

DISCINT and the I/O library kernel (IODECLARATIONS, etc.)

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1983.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S     RESTRICTED RIGHTS LEGEND
11:S
12:S     Use, duplication, or disclosure by the Government
13:S     is subject to restrictions as set forth in
14:S     paragraph (b) (3) (B) of the Rights in Technical
15:S     Data and Computer Software clause in
16:S     DAR 7-104.9(a).
17:S
18:S     HEWLETT-PACKARD COMPANY
19:S     0 Fort Collins, Colorado      *)
20:S
21:S
22:S     0 $MODCAL ON$
23:S     0 $PARTIAL_EVAL ON$
24:S     0 $STACKCHECK ON$
25:S     0 $RANGE OFF$
26:S     0 $DEBUG OFF$
27:S     0 $OVFLCHECK OFF$

```

```

28:D     0 $PAGE$
29:D
30:D     0 (*****
31:D     0 (*)
32:D     0 (* NOT RELEASED VERSION FOR 3.0 *)
33:D     0 (*)
34:D     0 (*****
35:D     0 (*)
36:D     0 (*)
37:D     0 (*) IOLIB DISCINT_DRIVERS (98625 Disc Interface) (*)
38:D     0 (*)
39:D     0 (*)
40:D     0 (*****
41:D     0 (*)
42:D     0 (*)
43:D     0 (*) library - IOLIB (*)
44:D     0 (*) name - DISCINT_DRIVERS (*)
45:D     0 (*) module(s) - di_drv (*)
46:D     0 (*) - discint (*)
47:D     0 (*)
48:D     0 (*) author - (*)
49:D     0 (*) phone - (*)
50:D     0 (*)
51:D     0 (*) date - May 5, 1982 (*)
52:D     0 (*) update - Aug 10, 1983 (*)
53:D     0 (*) release - ?????????? (*)
54:D     0 (*)
55:D     0 (*) source - IOLIB:DI_DRV.TEXT (*)
56:D     0 (*) object - IOLIB:DI_DRV.CODE (*)
57:D     0 (*)
58:D     0 (*****

```



```

59:D 0 $PAGES
60:D 0 (*****
61:D 0 *
62:D 0 *
63:D 0 * This is the source code for an external procedures library
64:D 0 * to be used for general purpose interfacing on the HP 9826.
65:D 0 *
66:D 0 * The library consists of 3 primary sets of modules -
67:D 0 *
68:D 0 * 1. KERNEL modules
69:D 0 * 2. driver modules
70:D 0 * 3. IOLIB modules
71:D 0 *
72:D 0 * The KERNEL modules consist of the following modules -
73:D 0 *
74:D 0 * 1. iodeclarations ( contains static r/w space )
75:D 0 * 2. iocomasm
76:D 0 * 3. general_0 ( initialization & low level
77:D 0 * routines like ioread/iowrite)
78:D 0 * The KERNEL modules also have an executable program segment
79:D 0 * that gets executed at the time it is loaded. This program
80:D 0 * initializes the static read/write memory. This program also
81:D 0 * allocates the temporary storage for any card that exists -
82:D 0 * independent of whether there is or is not a driver for it.
83:D 0 *
84:D 0 * The driver modules consist of the actual assembly or PASCAL
85:D 0 * routines that deal with a specific interface card. There is
86:D 0 * also an executable program segment for each driver module.
87:D 0 * This program searches the select code table in the static r/w
88:D 0 * initialized by the KERNEL general_0 module for all select codes
89:D 0 * that have the right interface card ( HPLIB drivers will search
90:D 0 * for the 98624 interface ). This program will then set up the
91:D 0 * driver tables to point to the correct drivers.
92:D 0 *
93:D 0 * The rest of the IOLIB modules are high-level modules that are
94:D 0 * used by an end user in his/her application program.
95:D 0 *
96:D 0 * The KERNEL and some set of driver modules will exist in the
97:D 0 * SYSTEM.INITLIB file as object code ( not EXPORT text ). The
98:D 0 * export text will reside on the SYSTEM.LIBRARY file. The rest
99:D 0 * of the library will reside on the SYSTEM.LIBRARY.
100:D 0 *
101:D 0 (*****

```

```

102:D 0 $PAGES
103:D 0 (*****
104:D 0 *
105:D 0 *
106:D 0 * REFERENCES :
107:D 0 *
108:D 0 *
109:D 0 * 1. 9826 I/O Designers Guide ( ----- )
110:D 0 *
111:D 0 * 2. 68000 Manual ( Motorola )
112:D 0 *
113:D 0 * 3. Pascal alpha site ERS ( ----- )
114:D 0 *
115:D 0 * 4. Pascal I/O Library ERS ( ----- )
116:D 0 *
117:D 0 * 5. 9826 HPL EIO & IOD listings ( ----- )
118:D 0 *
119:D 0 * 6. 9826 HPL Misc. I/O Doc. ( ----- )
120:D 0 *
121:D 0 * 7. 9826 card documentation ( Mfg. Specs. )
122:D 0 *
123:D 0 * 8. Pascal I/O Library IRS ( ----- )
124:D 0 *
125:D 0 *
126:D 0 (*****

```



```

127:D      0 $PAGES
128:D      0 PROGRAM discint_initialize ( INPUT , OUTPUT );
129:S
130:S
131:S      ( This module has a program segment so that there is
132:S      an executable entry point into the module.
133:D      1      At INITLIB time this program is executed. )

```

```

134:D      1 $PAGES
135:D      1 EXTERNAL MODULE extdi;
136:S      {
137:S          date    05/03/82
138:S          update  08/10/83
139:S
140:S          purpose This module is a declaration of the importation text for
141:S          the external drivers.
142:S
143:S          note    The assembly language code that is imported needs to be
144:S          called 'extdi'. The routines need to be called
145:S          'extdi_@@@' - 'edi_init' referenced below would be
146:S          extdi_edi_init. 'edi' refers to external disc interface.
147:D      1      }
148:S
149:D      1 IMPORT $SEARCH 'IDLIB:KERNEL.CODE'$
150:D      1      sysglobals ;
151:D      1      iodeclarations ;
152:S
153:D      1 EXPORT
154:S
155:D      1 PROCEDURE edi_init ( temp : ANYPTR );
156:D      1 PROCEDURE edi_isr ( temp : PISRB );
157:D      1 PROCEDURE edi_rdb ( temp : ANYPTR ; VAR x : CHAR );
158:D      1 PROCEDURE edi_wtb ( temp : ANYPTR ; val : CHAR );
159:D      1 PROCEDURE edi_rdw ( temp : ANYPTR ; VAR x : io_word );
160:D      1 PROCEDURE edi_wtw ( temp : ANYPTR ; val : io_word );
161:D      1 PROCEDURE edi_rds ( temp : ANYPTR ; reg : io_word ;
162:D      2      VAR x : io_word );
163:D      1 PROCEDURE edi_wtc ( temp : ANYPTR ; reg : io_word ;
164:D      2      val : io_word );
165:D      1 PROCEDURE edi_tfr ( temp : ANYPTR ; bcb : ANYPTR );
166:D      1 PROCEDURE edi_send ( temp : ANYPTR ; val : CHAR );
167:D      1 PROCEDURE edi_ppoll ( temp : ANYPTR ; VAR x : CHAR );
168:D      1 PROCEDURE edi_clr ( temp : ANYPTR ; line : io_bit );
169:D      1 PROCEDURE edi_set ( temp : ANYPTR ; line : io_bit );
170:D      1 PROCEDURE edi_test ( temp : ANYPTR ; line : io_bit ;
171:D      2      VAR x : BOOLEAN );
172:D      1 PROCEDURE edi_end ( temp : ANYPTR ; VAR x : BOOLEAN );
173:S
174:D      1 END; { of extdi }

```



```

175:D      1 $PAGE$
176:S
177:S
178:D      1 MODULE init_discint;
179:S      {
180:S          date    05/03/82
181:S          update  08/10/83 vers. number
182:S
183:S          purpose This module initializes the DISCINT drivers.
184:S
185:D      1      }
186:S
187:D      1 IMPORT   iodeclarations ;
188:S
189:S
190:D      1 EXPORT
191:D      1      VAR
192:D      -120 1      discint_drivers : drv_table_type;
193:S
194:S
195:D      -120 1  PROCEDURE io_init_discint;
196:S
197:S
198:D      -120 1 IMPLEMENT
199:S
200:D      -120 1  IMPORT   sysglobals ,
201:D      -120 1          isr ,
202:D      -120 1          general_0 ,
203:D      -120 1          extdi ;
204:S
205:S
206:D      1  PROCEDURE io_init_discint;
207:D      2  VAR
208:D      -2  2      io_isc      : type_isc;
209:D      -6  2      dummy      : INTEGER;
210:D      -8  2      io_lvl     : io_byte;
211:C      2  BEGIN
212:S
213:C      2      io_revid := io_revid + ' DI3.0';
214:S
215:C      2      { set up the driver tables }
216:S
217:C      2      WITH discint_drivers DO BEGIN
218:C      2          discint_drivers := dummy_drivers;
219:C      3          iod_init := edi_init;
220:C      3          iod_isr  := edi_isr;
221:C      3          iod_rdb  := edi_rdb;
222:C      3          iod_wtb  := edi_wtb;
223:C      3          iod_rdw  := edi_rdw;
224:C      3          iod_wtw  := edi_wtw;
225:C      3          iod_rds  := edi_rds;
226:C      3          iod_wtc  := edi_wtc;
227:C      3          iod_end  := edi_end;
228:C      3          iod_tfr  := edi_tfr;
229:C      3          iod_send := edi_send;
230:C      3          iod_ppoll := edi_ppoll;
231:C      3          iod_set  := edi_set;
232:C      3          iod_clr  := edi_clr;
233:C      3          iod_test := edi_test;
234:C      3      ENO; { of WITH }

```

```

235:S
236:S
237:C      2      { set up drivers for the interfaces }
238:S
239:C      2      FOR io_isc:=iomisc TO iomaxisc DO
240:C      3      WITH isc_table[io_isc] DO BEGIN
241:S
242:C      4          IF card_id = hp9862S
243:C      5          THEN BEGIN
244:S
245:C      5              io_drv_ptr:=ADDR(discint_drivers);
246:S
247:C      5              WITH io_drv_ptr^, io_tmp_ptr^ DO BEGIN
248:C      6                  io_lvl:=((io_read_byte(io_isc,3) DIV 16) MOD 4)+3;
249:S
250:C      6                  { if the card exists then link in an ISR for it }
251:C      6                  { ??? - what happens if an ISR fires during init }
252:S
253:C      6                  IF myisrib.INTREGADDR <> NIL
254:C      7                  THEN BEGIN
255:C      7                      { remove any existent isr }
256:C      7                      ISRUNLINK(io_lvl, { interrupt level }
257:C      7                      addr(myisrib)); { ptr to the isrib }
258:C      7                      ENO; { of IF }
259:S
260:C      6                      PERMISRLINK(iod_isr, { isr
261:C      6                      ANYPTR(INTEGER(card_ptr)+3), { int reg addr }
262:C      6                      192, { int reg mask }
263:C      6                      192, { int reg value }
264:C      6                      io_lvl, { int level }
265:C      6                      ADDR(myisrib)); { isrib info }
266:C      6                      ENO; { of WITH // DO BEGIN }
267:S
268:C      5                      ENO; { of IF card_id }
269:S
270:C      4                      ENO; { of FOR io_isc WITH isc_table[io_isc] BEGIN }
271:S
272:S
273:C      2      { call the actual driver initialization }
274:S
275:C      { this is separate from the set up stuff in case
276:C      there are 2 or more cards connected - and generate
277:C      an isr }
278:S
279:C      2      FOR io_isc:=iomisc TO iomaxisc DO
280:C      3      WITH isc_table[io_isc] DO
281:C      4      IF card_id = hp9862S
282:C      5      THEN BEGIN
283:C      5          CALL(io_drv_ptr^.iod_init , io_tmp_ptr);
284:C      5      ENO; { of WITH IF }
285:S
286:S
287:C      2  END; { of io_init_discint }
288:S
289:C      1 END; { of MODULE init_discint }

```



```
240:C      : $PAGE$
241:S
242:D      : IMPORT   init_discount ,
243:D      :          loader ;
244:S                                     { 367 TM 9/22/82 }
245:S
246:C      : BEGIN
247:C      :   to_init_discount;
248:C      :   MARKUSER;
249:C      :   END. { of discount_initialize }
250:S
251:S
```

No errors. No warnings.

***** Nonstandard language features enabled *****

DISCHPIB

Description

DISCHPIB is the common message-oriented HP-IB protocol interface used by the Amigo and CS80 mass storage transfer methods.

Requirements

IODECLARATIONS, HPIB or DISC_INTF

Optionally: DMA

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1983.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S     RESTRICTED RIGHTS LEGEND
11:S
12:S     Use, duplication, or disclosure by the Government
13:S     is subject to restrictions as set forth in
14:S     paragraph (b) (3) (B) of the Rights in Technical
15:S     Data and Computer Software clause in
16:S     DAR 7-104.9(a).
17:S
18:S     HEWLETT-PACKARD COMPANY
19:S     0 Fort Collins, Colorado      *)
20:S
21:S
22:D     0 $modcal, debug off, range off, ovfcheck off, stackcheck off, iocheck off$
23:D     0 $search 'DRVASHM', ;IOLIB:KERNEL'S
24:D
25:S
26:D     0 program init_bkgnd;
27:D
28:D     1 module bkgnd;
29:D
30:D     1 import
31:D     1 sysglobals, misc, iodeclarations;
32:S
33:D     1 export
34:D     1 type
35:D     1   uep_type = (unit entry pointer)
36:D     1   ^unitentry;
37:D     1   bi_type = (background info)
38:D     1   record
39:D     1     iores           : iorsltd;      {ioresult
40:D     1     set_in_use      : boolean;      {allocation flag
41:D     1     async           : boolean;      {overlapped transfer flag
42:D     1     feot           : eotproc;      {end of transfer procedure
43:D     1     fibptr         : fibp;         {file information block ptr
44:D     1     read_operation  : boolean;      {transfer direction flag
45:D     1     buffered_transfer : boolean;    {amigo driver flag
46:D     1     xfr_chain_semaphore : boolean;  {driver semaphore
47:D     1     bx_tries       : shortint;     {number of previous tries
48:D     1     bx_strt_rcrd    : integer;      {record address
49:D     1     bx_bufptr       : charptr;      {R/W address pointer
50:D     1     bx_length       : integer;      {total transfer byte count
51:D     1     bx_tfr_length   : integer;      {intermediate tfr byte count
52:D     1     bdx_chain_semaphore : boolean;  {data transfer semaphore
53:D     1     bdx_pre_eoi     : boolean;      {premature eoi flag
54:D     1     bdx_nb         : integer;      {data transfer byte count
55:D     1     bdx_proc        : io_proc;     {data transfer completion proc
56:D     1     buf_info       : buf_info_type; {as defined by the iolibary
57:D     1   end;
58:D     1   bip_type = ^bi_type;

```

```

59:D     1 $page$
60:S
61:D     1 procedure initialize_bkgnd;
62:D     1 procedure allocate_bkgnd_info (uep: uep_type);
63:D     1 procedure deallocate_bkgnd_info (uep: uep_type);
64:D     1 procedure abort_bkgnd_process (uep: uep_type);
65:D     1 procedure ioresc_bkgnd (uep: uep_type; ior: iorsltd);
66:D     1 function unit_busy (uep: uep_type): boolean;
67:D     1 procedure unit_wait (uep: uep_type);
68:S
69:D     1 implement (bkgnd)
70:S
71:D     1 const
72:D     1   n = 3; {number of bkgnd info sets}
73:S
74:D     1 type
75:D     1   bia_type = array[0..n-1] of bi_type;
76:S
77:D     1 var
78:D     -4 1   bia_ptr: ^bia_type;
79:D     -6 1   index: shortint;
80:S
81:S
82:D     1 procedure initialize_bkgnd;
83:D     2   var
84:D     -2 2   local_index: shortint;
85:D     2   begin {initialize_bkgnd}
86:D     2     if bia_ptr=nil then
87:D     2       new(bia_ptr);
88:D     2     for local_index := 0 to n-1 do
89:D     2       bia_ptr^[local_index].set_in_use := false;
90:D     2     index := 0;
91:D     2   end; {initialize_bkgnd}
92:S
93:S
94:D     1 function bip_valid(bip: bip_type): boolean;
95:D     2   var
96:D     -2 2   local_index: shortint;
97:D     2   begin {bip_valid}
98:D     2     bip_valid := false;
99:D     2     for local_index := 0 to n-1 do
100:D     3     if bip=addr(bia_ptr^[local_index]) then
101:D     4       bip_valid := true;
102:D     2   end; {bip_valid}

```



```

103:D  -6 1 $page$
104:S
105:D 1 procedure allocate_bkgnd_info(uep: uep_type);
106:D 2 var
107:D 3   bip: bip_type;
108:D 4   begin {allocate_bkgnd_info}
109:C 5     lockup;
110:C 6     repeat
111:C 7       bip := addr(bia_ptr^[index]);
112:C 8       index := index+1;
113:C 9       if index>=n then index := 0;
114:C10      until not bip^.set_in_use;
115:C11      uep^.dvrtemp := integer(bip);
116:C12      with bip^ do
117:C13      begin
118:C14        iores := ioreerror;
119:C15        set_in_use := true;
120:C16        async := false;
121:C17      end; {with}
122:C18    end; {allocate_bkgnd_info}
123:S
124:D 1 procedure deallocate_bkgnd_info(uep: uep_type);
125:D 2 var
126:D 3   saved_ioreresult: integer;
127:D 4   begin {deallocate_bkgnd_info}
128:C 5     if bip_valid(bip_type(uep^.dvrtemp)) then
129:C 6       with bip_type(uep^.dvrtemp)^ do
130:C 7       begin
131:C 8         set_in_use := false;
132:C 9         uep^.dvrtemp := ord(iores);
133:C10        lockdown;
134:C11        if async then {call the eot procedure}
135:C12        begin
136:C13          saved_ioreresult := ioreresult;
137:C14          ioreresult := uep^.dvrtemp;
138:C15          call(feot, fibptr);
139:C16          ioreresult := saved_ioreresult;
140:C17        end; {if}
141:C18      end; {with}
142:C19    end; {deallocate_bkgnd_info}
143:S
144:D 1 procedure abort_bkgnd_process(uep: uep_type);
145:D 2 begin {abort_bkgnd_process}
146:C 3   if escapecode<=-10 then {prevent any eot procedure call while deallocating}
147:C 4     bip_type(uep^.dvrtemp)^.async := false;
148:C 5     deallocate_bkgnd_info(uep);
149:C 6     if escapecode<=-10 then escape(escapecode);
150:C 7   end; {abort_bkgnd_process}
151:S
152:D 1 procedure ioresc_bkgnd(uep: uep_type; ior: iorsltwd);
153:D 2 begin {ioresc_bkgnd}
154:C 3   if bip_valid(bip_type(uep^.dvrtemp)) then
155:C 4     bip_type(uep^.dvrtemp)^.iores := ior;
156:C 5     escape(-10);
157:C 6   end; {ioresc_bkgnd}
158:C
159:C
160:C

```

```

161:D  -6 1 $page$
162:S
163:D 1 function unit_busy(uep: uep_type): boolean;
164:D 2 begin {unit_busy}
165:C 3   if bip_valid(bip_type(uep^.dvrtemp)) then
166:C 4     unit_busy := true;
167:C 5   else
168:C 6     begin
169:C 7       unit_busy := false;
170:C 8       ioreresult := uep^.dvrtemp;
171:C 9     end; {else}
172:C10   end; {unit_busy}
173:S
174:D 1 procedure unit_wait(uep: uep_type);
175:D 2 begin {unit_wait}
176:C 3   while unit_busy(uep) do {nothing};
177:C 4   end; {unit_wait}
178:S
179:D 1 end; {bkgnd}
180:S
181:C

```



```

182:D 1 $page$
183:D 1 $copyright 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'$
184:S
185:S
186:D 1 module dischPIB;
187:S
188:D 1 import
189:D 1 sysglobals, iodeclarations, drvasm, bkgnd;
190:S
191:D 1 export
192:D 1 function Simon_no_DMA (uep: uep_type): boolean;
193:D 1 function Simon_DMA (uep: uep_type): boolean;
194:D 1 procedure HPIBcheck_sc (uep: uep_type);
195:D 1 procedure HPIBwait_for_ppol (uep: uep_type);
196:D 1 procedure HPIBshort_msge_out (uep: uep_type; sec: byte; bp: charptr; nb: shortint);
197:D 1 procedure HPIBamigo_clear (uep: uep_type);
198:D 1 procedure HPIBshort_msge_in (uep: uep_type; sec: byte; bp: charptr; nb: shortint);
199:D 1 function HPIBamigo_identify (uep: uep_type): shortint;
200:D 1 procedure HPIBget_amigo_ident (uep: uep_type; var ident: shortint);
201:D 1 procedure HPIBupon_ppol_resp (uep: uep_type; proc: io_proc);
202:D 1 procedure HPIBupon_dxfr_comp (uep: uep_type; sec: byte; bp: charptr; nb: integer; proc: io_proc);
203:D
204:D 1 implement {dischPIB}
205:S
206:S
207:D 1 const
208:D 1 standard_tc = 5000; {standard byte timeout value milliseconds}
209:D 1 short_tc = 25; {short byte timeout value milliseconds}
210:D 1 SDC = 4; {selective device clear}
211:D 1 LAG_base = 32; {listen address group base}
212:D 1 TAG_base = 64; {talk address group base}
213:D 1 SCG_base = 96; {secondary command group base}
214:S
215:S
216:D 1 procedure delay_timer (microsec_value: integer); external;
217:S
218:S
219:D 1 procedure confirm_timeout_and_reset_card (uep: uep_type);
220:D 2 begin {confirm_timeout_and_reset_card}
221:D 2 if {escapecode<>ioescApeCode} or {ioesc<>uep^.sc} then escape(escapecode);
222:D 2 if ioesc_result<>ioesc_timeout then ioresc_bkgnd(uep, znodevice);
223:D 2 with isc_table[uep^.sc] do
224:D 3 call(io_drv_ptr^.iod_init, io_tmp_ptr);
225:D 2 ioresc_bkgnd(uep, ztimeout);
226:D 2 end; {confirm_timeout_and_reset_card}
227:S

```

```

228:D 1 $page$
229:S
230:D 1 function Simon_no_DMA (uep: uep_type): boolean;
231:C 2 begin {Simon_no_DMA}
232:C 2 with isc_table[uep^.sc] do
233:C 3 Simon_no_DMA := (card_id=hp98625) and not dma_here;
234:C 2 end; {Simon_no_DMA}
235:S
236:S
237:D 1 function Simon_DMA (uep: uep_type): boolean;
238:C 2 begin {Simon_DMA}
239:C 2 with isc_table[uep^.sc] do
240:C 3 Simon_DMA := (card_id=hp98625) and dma_here;
241:C 2 end; {Simon_DMA}
242:S
243:S
244:D 1 procedure HPIBcheck_sc (uep: uep_type);
245:C 2 begin {HPIBcheck_sc}
246:C 2 with isc_table[uep^.sc] do
247:C 3 begin
248:C 3 if card_type<>hpib_card then
249:C 4 ioresc_bkgnd(uep, znodevice);
250:C 3 with io_tmp_ptr do
251:C 4 while (in_bufptr<>nil) or (out_bufptr<>nil) do {nothing};
252:C 3 end; {with}
253:C 2 end; {HPIBcheck_sc}
254:S
255:S
256:D 1 procedure HPIBwait_for_ppol (uep: uep_type);
257:D 2 var
258:D -2 pprb: packed array[0..7] of boolean; {parallel poll response byte}
259:D 2 begin {HPIBwait_for_ppol}
260:D 2 try
261:D 3 with isc_table[uep^.sc], io_drv_ptr do
262:D 4 repeat
263:D 5 call(iod_ppoll, io_tmp_ptr, charptr(addr(pprb))^);
264:D 5 until pprb[uep^.ba];
265:D 3 recover
266:D 3 confirm_timeout_and_reset_card(uep);
267:D 2 end; {HPIBwait_for_ppol}

```



```

268:O 1 $page$
269:S
270:D
271:D
272:D -1 2
273:C 2
274:C 2
275:C 3
276:C 3
277:C 3
278:C 3
279:C 3
280:C 3
281:C 4
282:C 3
283:C 3
284:C 3
285:C 2
286:C 2
287:D 1
288:D 2
289:D -4 2
290:C 2
291:C 2
292:C 3
293:C 3
294:C 3
295:C 4
296:C 4
297:C 5
298:C 5
299:C 5
300:C 5
301:C 5
302:C 4
303:C 4
304:C 4
305:C 4
306:C 3
307:C 3
308:C 2
309:S
310:D 1
311:D 2
312:D -4 2
313:C 2
314:C 2
315:C 3
316:C 3
317:C 3
318:C 4
319:C 4
320:C 4
321:C 4
322:C 4
323:C 4
324:C 3
325:C 3
326:C 2

1 procedure address_for_msge_out(var isc_te: isc_table_type; ba, sec: byte; tc: integer);
2 var
3   dummy_char: char;
4 begin {address_for_msge_out}
5   with isc_te, io_drv_ptr^ do
6   begin
7     call(iod_send, io_tmp_ptr, '?');
8     io_tmp_ptr^.timeOut := tc;
9     call(iod_send, io_tmp_ptr, chr((TAG_base+io_tmp_ptr^.addressed)));
10    call(iod_send, io_tmp_ptr, chr((LAG_base+ba)));
11    if card_id<>hp88625 then {delay to avoid Chinook bug}
12      delay_timer(85 {microseconds});
13    call(iod_send, io_tmp_ptr, chr((SCG_base+sec)));
14    call(iod_ppoll, io_tmp_ptr, dummy_char); {enforce timeout}
15  end; {with}
16 end; {address_for_msge_out}

1 procedure HPIBshort_msge_out(uep: uep_type; sec: byte; bp: charptr; nb: shortint);
2 var
3   isc_te_ptr: ^isc_table_type;
4 begin {HPIBshort_msge_out}
5   try
6     isc_te_ptr := addr(isc_table[uep^.sc]);
7     address_for_msge_out(isc_te_ptr, uep^.ba, sec, standard_tc);
8     with isc_te_ptr^, io_drv_ptr^ do
9     begin
10      while nb>1 do
11      begin
12        call(iod_wtb, io_tmp_ptr, bp^);
13        bp := addr(bp^, 1);
14        nb := nb-1;
15      end; {while}
16      call(iod_set, io_tmp_ptr, ord(eoi_line));
17      call(iod_wtb, io_tmp_ptr, bp^);
18      call(iod_send, io_tmp_ptr, '?');
19    end; {with}
20    recover
21    confirm_timeout_and_reset_card(uep);
22 end; {HPIBshort_msge_out}

1 procedure HPIBamigo_clear(uep: uep_type);
2 var
3   isc_te_ptr: ^isc_table_type;
4 begin {HPIBamigo_clear}
5   try
6     isc_te_ptr := addr(isc_table[uep^.sc]);
7     address_for_msge_out(isc_te_ptr, uep^.ba, 16, standard_tc);
8     with isc_te_ptr^, io_drv_ptr^ do
9     begin
10      call(iod_set, io_tmp_ptr, ord(eoi_line));
11      call(iod_wtb, io_tmp_ptr, chr(0)); {disable parity check}
12      call(iod_send, io_tmp_ptr, chr(SDC));
13      call(iod_send, io_tmp_ptr, '?');
14    end; {with}
15    recover
16    confirm_timeout_and_reset_card(uep);
17 end; {HPIBamigo_clear}

```

```

327:D 1 $page$
328:S
329:D
330:D
331:D -1 2
332:C 2
333:C 2
334:C 3
335:C 3
336:C 3
337:C 3
338:C 3
339:C 3
340:C 4
341:C 3
342:C 3
343:C 3
344:C 4
345:C 3
346:C 2
347:S
348:D
349:D
350:D
351:C 2
352:C 2
353:C 2
354:C 2
355:S
356:D
357:D
358:D
359:D -4 2
360:D -5 2
361:C 2
362:C 3
363:C 3
364:C 3
365:C 3
366:C 4
367:C 4
368:C 5
369:C 5
370:C 5
371:C 5
372:C 5
373:C 5
374:C 5
375:C 4
376:C 4
377:C 4
378:C 3
379:C 3
380:C 2

1 procedure address_for_msge_in(var isc_te: isc_table_type; ba, sec: byte; tc: integer);
2 var
3   dummy_char: char;
4 begin {address_for_msge_in}
5   with isc_te, io_drv_ptr^ do
6   begin
7     call(iod_send, io_tmp_ptr, '?');
8     io_tmp_ptr^.timeOut := tc;
9     call(iod_send, io_tmp_ptr, chr((LAG_base+io_tmp_ptr^.addressed)));
10    call(iod_send, io_tmp_ptr, chr((TAG_base+ba)));
11    if card_id<>hp88625 then {delay to avoid Chinook bug}
12      delay_timer(85 {microseconds});
13    call(iod_send, io_tmp_ptr, chr((SCG_base+sec)));
14    call(iod_ppoll, io_tmp_ptr, dummy_char); {enforce timeout}
15    if card_id<>hp88625 then {delay to avoid Chinook bug}
16      delay_timer(85 {microseconds});
17  end; {with}
18 end; {address_for_msge_in}

1 procedure premature_eoi(uep: uep_type);
2 begin {premature_eoi}
3   with isc_table[uep^.sc] do
4     call(io_drv_ptr^.iod_send, io_tmp_ptr, '?');
5   ioresc_bkgnd(uep, zbadhardware); {all "expected" premature eoi's have to be trapped}
6 end; {premature_eoi}

1 procedure HPIBshort_msge_in(uep: uep_type; sec: byte; bp: charptr; nb: shortint);
2 var
3   isc_te_ptr: ^isc_table_type;
4   eoi_set: boolean;
5 begin {HPIBshort_msge_in}
6   try
7     isc_te_ptr := addr(isc_table[uep^.sc]);
8     address_for_msge_in(isc_te_ptr, uep^.ba, sec, standard_tc);
9     with isc_te_ptr^, io_drv_ptr^ do
10     begin
11      while nb>1 do
12      begin
13        call(iod_rdb, io_tmp_ptr, bp^);
14        call(iod_end, io_tmp_ptr, eoi_set);
15        if eoi_set then premature_eoi(uep);
16        bp := addr(bp^, 1);
17        nb := nb-1;
18      end; {while}
19      call(iod_rdb, io_tmp_ptr, bp^);
20      call(iod_send, io_tmp_ptr, '_');
21    end; {with}
22    recover
23    confirm_timeout_and_reset_card(uep);
24 end; {HPIBshort_msge_in}

```



```

361:D 1 $page$
362:S
363:D 1 function HPIBamigo_identify(uep: uep_type): shortint;
364:D 2 var
365:D -4 2   isc_te_ptr: ^isc_table_type;
366:D -4 2   ident: (the two identify bytes)
367:D -4 2   record case integer of
368:D -4 2     0: (word: shortint);
369:D -4 2     1: (upper_char, lower_char: char);
370:D -4 2   end;
371:D -7 2   eoi_set: boolean;
372:D 2   begin (HPIBamigo_identify)
373:C 2   try
374:C 3     isc_te_ptr := addr(isc_table[uep^.sc]);
375:C 3     address_for_msge_in(isc_te_ptr^, {"ba"} 31, {"sec"} uep^.ba, short_tc);
376:C 3     with isc_te_ptr^, io_drv_ptr^ do
377:C 4       begin
378:C 4         call(io_drv_ptr, io_tmp_ptr, ident.upper_char);
379:C 4         call(io_drv_ptr, io_tmp_ptr, eoi_set);
380:C 4         if eoi_set then premature_eoi(uep);
381:C 4         call(io_drv_ptr, io_tmp_ptr, ident.lower_char);
382:C 4         call(io_drv_ptr, io_tmp_ptr, chr(TAG_base+io_tmp_ptr^.addressed));
383:C 4       end; (with)
384:C 3     recover
385:C 3     confirm_timeout_and_reset_card(uep);
386:C 2   HPIBamigo_identify := ident.word;
387:C 2   end; (HPIBamigo_identify)
388:S
389:S
390:D 1 procedure HPIBget_amigo_ident(uep: uep_type; var ident: shortint);
391:D 2 begin (HPIBget_amigo_ident)
392:D 2   ident := HPIBamigo_identify(uep);
393:D 2   end; (HPIBget_amigo_ident)
394:S
395:S
396:D 1 procedure HPIBupon_ppol_resp(uep: uep_type; proc: io_proc);
397:D 2 {
398:D 2   NOTE: when SIMON drivers become available, this routine needs to
399:D 2   be modified to utilize the "interrupt on parallel poll response"
400:D 2   capability of SIMON. However, until then, this will have to do.
401:D 2 }
402:D 2 var
403:D -2 2   pprb: packed array[0..7] of boolean;      (parallel poll response byte)
404:D 2   begin (HPIBupon_ppol_resp)
405:C 2   try
406:C 3     with isc_table[uep^.sc], io_drv_ptr^ do
407:C 4       repeat
408:C 4         call(io_drv_ptr, io_tmp_ptr, charptr(addr(pprb))^);
409:C 4       until pprb[uep^.ba];
410:C 3     recover
411:C 3     confirm_timeout_and_reset_card(uep);
412:C 2   call(proc, uep);
413:C 2   end; (HPIBupon_ppol_resp)

```

```

434:D 1 $page$
435:S
436:D 1 procedure enter_bdx_chain(uep: uep_type); forward;
437:D 1 procedure initiate_transfer(uep: uep_type); forward;
438:D 1 procedure upon_transfer_complete(uep: anyptr); forward;
439:S
440:D 1 procedure HPIBupon_dxfr_comp(uep: uep_type; sec: byte; bp: charptr; nb: integer; proc: io_proc);
441:D 2 var
442:D -4 2   isc_te_ptr: ^isc_table_type;
443:D -6 2   t_dir: dir_of_tfr;
444:D 2   begin (HPIBupon_dxfr_comp)
445:C 2   try
446:C 3     isc_te_ptr := addr(isc_table[uep^.sc]);
447:C 3     with bdp_type(uep^.drv_tmp)^ do
448:C 4       begin
449:C 4         if read_operation then
450:C 5           begin
451:C 5             address_for_msge_in(isc_te_ptr^, uep^.ba, sec, standard_tc);
452:C 5             t_dir := to_memory;
453:C 5           end (then)
454:C 5         else
455:C 5           begin
456:C 5             address_for_msge_out(isc_te_ptr^, uep^.ba, sec, standard_tc);
457:C 5             t_dir := from_memory;
458:C 5           end (else)
459:C 4       end;
460:C 4       bdx_nb := nb;
461:C 4       bdx_proc := proc;
462:C 4       with isc_te_ptr^, buf_info do
463:C 5         begin
464:C 5           drv_tmp_ptr := io_tmp_ptr;
465:C 5           active_isc := no_isc;
466:C 5           { act_tfr is set by the driver }
467:C 5           if dma_here
468:C 6             then usr_tfr := overlap_FASTEST {DMA, or BURST FHS if both channels are busy}
469:C 6             else usr_tfr := serial_FHS; {unlike BURST FHS, won't lock out interrupts}
470:C 5           b_w_mode := false;
471:C 5           { end_mode is setup in initiate_transfer }
472:C 5           direction := t_dir;
473:C 5           term_char := -1;
474:C 5           { term_count is setup in initiate_transfer }
475:C 5           buf_ptr := anyptr(bp);
476:C 5           buf_size := nb;
477:C 5           buf_empty := anyptr(bp);
478:C 5           buf_fill := anyptr(bp);
479:C 5           eot_proc.real_proc := upon_transfer_complete;
480:C 5           eot_parm := uep;
481:C 5           dma_priority := card_id=hp98625;
482:C 5         end; (with)
483:C 4       bdx_chain_semaphore := false;
484:C 4       enter_bdx_chain(uep);
485:C 4       end; (with)
486:C 3     recover
487:C 3     confirm_timeout_and_reset_card(uep);
488:C 2   end; (HPIBupon_dxfr_comp)

```



```

493:D      1 $page$
494:S
495:D      1 procedure enter_bdx_chain(uep: uep_type);
496:C      2 begin {enter_bdx_chain}
497:C      3   with bdp_type(uep^.dvrtemp)^ do
498:C      4     if not test_and_toggle(bdx_chain_semaphore) then
499:C      5       repeat
500:C      6         initiate_transfer(uep);
501:C      7         until test_and_toggle(bdx_chain_semaphore);
502:C      8   end; {enter_bdx_chain}
503:S
504:D      1 procedure initiate_transfer(uep: uep_type);
505:D      2 var
506:D      3   maximum_term_count: integer;
507:D      4   -1 begin {initiate_transfer}
508:C      5     with bdp_type(uep^.dvrtemp)^, isc_table[uep^.sc], buf_info do
509:C      6       begin
510:C      7         if (usr_tfr=serial_FHS) or (card_id=hp98625)
511:C      8           then maximum_term_count := maxInt { "no" limitation}
512:C      9           else maximum_term_count := 65536; {DMA hardware/9914 driver limitation}
513:C      4         if bdx_nb <= maximum_term_count
514:C      5           then term_count := bdx_nb
515:C      6           else term_count := maximum_term_count;
516:C      7         bdx_nb := bdx_nb - term_count;
517:C      8         end_mode := (direction=to_memory) or (bdx_nb=0);
518:C      9         call(io_drv_ptr^.iod_tfr, io_tmp_ptr, addr(buf_info));
519:C      3       end; {with}
520:C      2     end; {initiate_transfer}
521:S
522:D      1 procedure upon_transfer_complete(uep: anyptr);
523:D      2 var
524:D      3   unaddressing_char: char;
525:D      4   -1 begin {upon_transfer_complete}
526:C      5     with bdp_type(uep_type(uep)^(dvrtemp)^, isc_table[uep_type(uep)^(sc)], io_drv_ptr^, buf_info do
527:C      6       try
528:C      7         if direction=to_memory then {check for premature transfer termination}
529:C      8           if bdx_nb=0
530:C      9             then bdx_pre_eoi := term_count <> 0
531:C      6           else call(iod_end, io_tmp_ptr, bdx_pre_eoi)
532:C      7         else
533:C      8           bdx_pre_eoi := false;
534:C      9         if (bdx_nb > 0) and not bdx_pre_eoi then {re-initiate the transfer}
535:C      4           enter_bdx_chain(uep)
536:C      5         else {unaddress the bus and call the specified end-of-transfer procedure}
537:C      6           begin
538:C      7             if direction=to_memory
539:C      8               then unaddressing_char := ' ' {untalk}
540:C      9               else unaddressing_char := '7'; {unlisten}
541:C      4             call(iod_send, io_tmp_ptr, unaddressing_char);
542:C      5             call(bdx_proc, uep);
543:C      6           end; {else}
544:C      4         recover
545:C      5         confirm_timeout_and_reset_card(uep);
546:C      2       end; {upon_transfer_complete}
547:S
548:D      1 end; {dischpiB}
549:S

```

```

550:D      1 $page$
551:S
552:D      1 import
553:D      2 loader, bkgnd;
554:S
555:C      1 begin {init_bkgnd}
556:C      2   initialize_bkgnd; {allocate temp space}
557:C      3   markuser; {make temp space and modules permanent}
558:C      4   end. {init_bkgnd}
559:S
560:S
561:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

DMA_DRV

Description

DMA_DRV provides low-level driver support.

Usage

DMA_DRV is used with the 98620A/B DMA card.

Requirements

The I/O library kernel (IODECLARATIONS, etc.)

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*)
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1983.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S      RESTRICTED RIGHTS LEGEND
11:S
12:S      Use, duplication, or disclosure by the Government
13:S      is subject to restrictions as set forth in
14:S      paragraph (b) (3) (B) of the Rights in Technical
15:S      Data and Computer Software clause in
16:S      DAR 7-104.9(a).
17:S
18:S      HEWLETT-PACKARD COMPANY
19:D      0 Fort Collins, Colorado      *)
20:S
21:S
22:D      0 $MODCAL ON$
23:D      0 $PARTIAL_EVAL ON$
24:D      0 $STACKCHECK ON$
25:D      0 $RANGE OFF$
26:D      0 $DEBUG OFF$
27:D      0 $OVFLCHECK OFF$

```

```

28:D      0 $PAGE$
29:S
30:D      0 (*****
31:D      0 *
32:D      0 *      NOT RELEASED      VERSION      3.0
33:D      0 *
34:D      0 (*****
35:D      0 *
36:D      0 *
37:D      0 *      IOLIB      DMA_DRIVERS
38:D      0 *
39:D      0 *
40:D      0 (*****
41:D      0 *
42:D      0 *
43:D      0 *      library      - IOLIB
44:D      0 *      name      - DMA_DRIVERS
45:D      0 *      module(s) - init_dma
46:D      0 *
47:D      0 *      author      -
48:D      0 *      phone      -
49:D      0 *
50:D      0 *      date      - June 1 , 1981
51:D      0 *      update      - Aug 10 , 1983 - rev. number
52:D      0 *      release      - ?????????????
53:D      0 *
54:D      0 *      source      - IOLIB:DMA_DRV.TEXT
55:D      0 *      object      - IOLIB:DMA_DRV.CODE
56:D      0 *
57:D      0 (*****

```



```
58:D 0 $PAGES$
59:D 0 (*****
60:D 0 *
61:D 0 *
62:D 0 *      BUG FIX HISTORY      - after release 1.0
63:D 0 *
64:D 0 *
65:D 0 *      BUG #    BY / ON      LOC      DESCRIPTION
66:D 0 *      -----
67:D 0 *
68:D 0 *      367      09/22/82      dma_initialize Allow execute of driver
69:D 0 *                                     and have it install
70:D 0 *                                     itself in the system.
71:D 0 *
72:D 0 (*****
```

```
73:D 0 $PAGES$
74:D 0 (*****
75:D 0 *
76:D 0 *
77:D 0 *      This is the source code for an external procedures library
78:D 0 *      to be used for general purpose interfacing on the HP 9826.
79:D 0 *
80:D 0 *      The library consists of 3 primary sets of modules -
81:D 0 *
82:D 0 *          1.      KERNEL modules
83:D 0 *          2.      driver modules
84:D 0 *          3.      IOLIB modules
85:D 0 *
86:D 0 *      The KERNEL modules consist of the following modules -
87:D 0 *
88:D 0 *          1.      iodeclarations ( contains static r/w space )
89:D 0 *          2.      iocomasm
90:D 0 *          3.      general_0      ( initialization & low level
91:D 0 *                                routines like ioread/iowrite)
92:D 0 *
93:D 0 *      The KERNEL modules also have an executable program segment
94:D 0 *      that gets executed at the time it is loaded. This program
95:D 0 *      initializes the static read/write memory. This program also
96:D 0 *      allocates the temporary storage for any card that exists -
97:D 0 *      independent of whether there is or is not a driver for it.
98:D 0 *
99:D 0 *      The driver modules consist of the actual assembly or PASCAL
100:D 0 *      routines that deal with a specific interface card. There is
101:D 0 *      also an executable program segment for each driver module.
102:D 0 *
103:D 0 *      This program searches the select code table in the static r/w
104:D 0 *      initialized by the KERNEL general_0 module for all select codes
105:D 0 *      that have the right interface card ( HP1B drivers will search
106:D 0 *      for the 98624 interface ). This program will then set up the
107:D 0 *      driver tables to point to the correct drivers.
108:D 0 *
109:D 0 *      The rest of the IOLIB modules are high-level modules that are
110:D 0 *      used by an end user in his/her application program.
111:D 0 *
112:D 0 *      The KERNEL and some set of driver modules will exist in the
113:D 0 *      SYSTEM.INITLIB file as object code ( not EXPORT text ). The
114:D 0 *      export text will reside on the SYSTEM.LIBRARY file. The rest
115:D 0 *      of the library will reside on the SYSTEM.LIBRARY.
115:D 0 (*****
```



```
116:D      0 $PAGE$
117:D      0 (*****
118:D      0 (*
119:D      0 (*
120:D      0 (*      REFERENCES :
121:D      0 (*
122:D      0 (*
123:D      0 (*      1.  9826 I/O Designers Guide      ( ----- )
124:D      0 (*
125:D      0 (*      2.  68000 Manual      ( Motorola )
126:D      0 (*
127:D      0 (*      3.  Pascal alpha site ERS      ( ----- )
128:D      0 (*
129:D      0 (*      4.  Pascal I/O Library ERS      ( ----- )
130:D      0 (*
131:D      0 (*      5.  9826 HPL EIO & IOD listings      ( ----- )
132:D      0 (*
133:D      0 (*      6.  9826 HPL Misc. I/O Doc.      ( ----- )
134:D      0 (*
135:D      0 (*      7.  9826 card documentation      ( Mfg. Specs. )
136:D      0 (*
137:D      0 (*      8.  Pascal I/O Library ISR      ( ----- )
138:D      0 (*
139:D      0 (*
140:D      0 (*****
```

```
141:D      0 $PAGE$
142:D      0 PROGRAM dma_initialize ( INPUT , OUTPUT );
```



```

143:D      1 $PAGE$
144:D      1 (*****
145:D      1 (*
146:D      1 (*
147:D      1 (*      DMA DRIVERS
148:D      1 (*
149:D      1 (*
150:D      1 (*****

```

```

151:D      1 $PAGE$
152:D      1 MODULE init_dma ;
153:S
154:S      {
155:S          date    09/23/81
156:S          update  10/04/82
157:S
158:S          purpose This module is a series of DMA procedures to be linked into
159:S                  the system isr poll routines. The module also contains the
160:S                  initialization code.
161:D      1      }
162:S
163:D      1 IMPORT $SEARCH 'IOLIB:KERNEL.CODE'$
164:D      1      sysglobals, iodeclarations ;
165:S
166:D      1 EXPORT
167:S
168:S
169:D      1 PROCEDURE dma_isr_0 ( an_isrib : PISRIB );
170:D      1 PROCEDURE dma_isr_1 ( an_isrib : PISRIB );
171:D      1 PROCEDURE io_init_dma;
172:S
173:S
174:D      1 IMPLEMENT
175:D      1
176:S
177:D      1 IMPORT isr,
178:D      1      general_0 ;
179:S
180:S
181:S
182:D      1 PROCEDURE dma_isr_0 ( an_isrib : PISRIB );
183:D      -2 2 VAR io_work_word : io_word;
184:C      2 BEGIN
185:C      2 IF dma_ch_0.dummy_pr = NIL
186:C      3 THEN BEGIN
187:C      3 { no procedure for dma isr - disable channel }
188:C      3 io_work_word:=ioread_word(3,0);
189:C      3 END
190:C      3 ELSE BEGIN
191:C      3 { call the procedure with a temp ptr }
192:C      3 CALL(dma_ch_0.real_proc,isc_table[dma_isc_0].io_tmp_ptr);
193:C      3 END; { of IF }
194:C      2 END; { of dma_isr_0 }
195:S
196:S
197:S
198:S
199:D      1 PROCEDURE dma_isr_1 ( an_isrib : PISRIB );
200:D      -2 2 VAR io_work_word : io_word;
201:C      2 BEGIN
202:C      2 IF dma_ch_1.dummy_pr = NIL
203:C      3 THEN BEGIN
204:C      3 { no procedure for dma isr - disable channel }
205:C      3 io_work_word:=ioread_word(3,0);
206:C      3 END
207:C      3 ELSE BEGIN
208:C      3 { call the procedure with a temp ptr }
209:C      3 CALL(dma_ch_1.real_proc,isc_table[dma_isc_1].io_tmp_ptr);
210:C      3 END; { of IF }

```



```

211:C      2  END; { of dma_isr_1 }
212:S
213:S
214:D      1  PROCEDURE io_init_dma;
215:D      -4  VAR dummy      : INTEGER;
216:C      2  BEGIN
217:S
218:C      2  io_revid := io_revid + ' D3.0';      { DMA revision added 2/5/82 TM }
219:S
220:C      2  { initialize the DMA variables }
221:S
222:C      2  dma_ch_0.dummy_pr := NIL;
223:C      2  dma_ch_0.dummy_sl := NIL;
224:C      2  dma_isc_0      := no_isc;
225:C      2  dma_ch_1.dummy_pr := NIL;
226:C      2  dma_ch_1.dummy_sl := NIL;
227:C      2  dma_isc_1      := no_isc;
228:S
229:S
230:S
231:C      2  { see if dma really exists }
232:S
233:C      2  TRY
234:C      3  dummy:=io_read_word(3,0);
235:C      3  dma_here:=TRUE;
236:C      3  isc_table[3].card_id:=hp98620;
237:C      3  RECOVER
238:C      3  BEGIN
239:C      4  IF escapecode=-12
240:C      4  THEN BEGIN
241:C      4  dma_here := FALSE;
242:C      4  END
243:C      4  ELSE BEGIN
244:C      4  { some other problem }
245:C      4  ESCAPE(ESCAPECODE);
246:C      4  END; { of IF }
247:C      3  END; { of RECOVER BEGIN }
248:S
249:S
250:C      2  { if the card has isrs then unlink the ISR }
251:S
252:C      2  IF dma_isrib0.INTREGADDR <> NIL THEN
253:C      3  ISRUNLINK(3, { interrupt level }
254:C      3  ADDR(dma_isrib0)); { ptr to the isrib }
255:S
256:C      2  IF dma_isrib1.INTREGADDR <> NIL THEN
257:C      3  ISRUNLINK(3, { interrupt level }
258:C      3  ADDR(dma_isrib1)); { ptr to the isrib }
259:S
260:S
261:C      2  { if the card exists then link in an ISR for it }
262:S
263:C      2  IF dma_here
264:C      3  THEN WITH isc_table[3] DO BEGIN
265:C      4  { dma if installed is always int. level 3 }
266:C      4  PERMISRLINK(dma_isr_0, { channel 0 isr }
267:C      4  ANYPTR(INTEGER(card_ptr)+7), { channel 0 addr }
268:C      4  2, { int reg mask }
269:C      4  2, { int reg value }
270:C      4  3, { int level }

```

```

271:C      4  ADDR(dma_isrib0)); { isrib 0 info }
272:C      4  PERMISRLINK(dma_isr_1, { channel 1 isr }
273:C      4  ANYPTR(INTEGER(card_ptr)+15), { channel 1 addr }
274:C      4  2, { int reg mask }
275:C      4  2, { int reg value }
276:C      4  3, { int level }
277:C      4  ADDR(dma_isrib1)); { isrib 1 info }
278:C      4  END; { of IF }
279:S
280:C      2  END; { of io_init_dma }
281:S
282:C      1  END; { of MODULE init_dma }

```



```
283:D      1 $PAGES
284:S
285:D      1 IMPORT      init_dma ,
286:D      1              LOADER ;
287:S
288:S
289:C      1 BEGIN
290:C      1   io_init_dma;
291:C      1   MARKUSER;
292:C      1 END. { of dma_initialize }
                { 367 TM 9/22/82 }
```

No errors. No warnings.

***** Nonstandard language features enabled *****

EDRIVER

Description

EDRIVER contains a utility library for driving the EPROM programming card.

Requirements

SYSGLOBALS, IODECLARATIONS, and ASM.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```
1: S (*
2: S
3: S (c) Copyright Hewlett-Packard Company, 1984.
4: S All rights are reserved. Copying or other
5: S reproduction of this program except for archival
6: S purposes is prohibited without the prior
7: S written consent of Hewlett-Packard Company.
8: S
9: S
10: S RESTRICTED RIGHTS LEGEND
11: S
12: S Use, duplication, or disclosure by the Government
13: S is subject to restrictions as set forth in
14: S paragraph (b) (3) (B) of the Rights in Technical
15: S Data and Computer Software clause in
16: S DAR 7-104.9(a)
17: S
18: S HEWLETT-PACKARD COMPANY
19: S 0 Fort Collins, Colorado *)
20: S
21: D 0 $SYSPROG$
22: D 0 $RANGE OFF$ $OEBUG OFF$
23: D 0 $MODULE EDRIVER;
24: D 1 IMPORT SYSGLOBALS, IODECLARATIONS, ASM;
25: D 1 EXPORT
26: D 1 TYPE
27: D 1 EPINFOREC = RECORD
28: D 1 BFAST : BOOLEAN;
29: D 1 EPSTART : INTEGER;
30: D 1 EPEND : INTEGER;
31: D 1 END;
32: S
33: D 1 EOCTYPE = (EREAD, EWRITE, ECWRITE, EBLANK, ECHECK);
34: D 1 EPEROR = (ENERROR, ENOTPROG, ENOEPROM, ECFAIL, EPFAIL, ENOTBLANK, EBADARGS);
35: S
36: D 1 FUNCTION EPROG( SCODE : INTEGER;
37: D 1 OP : EOCTYPE;
38: D 1 ANYVAR BUFFER : WINDOW;
39: D 1 BUFSIZE : INTEGER;
40: D 2 VAR POSIT : INTEGER): EPEROR;
41: S
42: D 1 FUNCTION EGETINFO( SCODE : INTEGER;
43: D 2 VAR INFO : EPINFOREC): EPEROR;
44: S
45: D 1 FUNCTION EINIT(SCODE : INTEGER): EPEROR;
46: S
47: D 1 FUNCTION EBRATE(SCODE : INTEGER;
48: D 2 BRATE : BOOLEAN): EPEROR;
49: S
50: D 1 IMPLEMENT
51: D 1 TYPE
52: D 1 EOCTYPE = (EINIT, EPREAD, EPWRITE, EPCWRITE, EPLANK, EPCHECK, EPBRATE, EPINFO);
53: D 1 STATREG= PACKED RECORD
54: D 1 B4 : BYTE; { PUT HERE TO MAKE THE COMPILER HAPPY }
55: D 1 PA0 : 0..15; { 7,6,5,4 }
56: D 1 FAST : BOOLEAN;
57: D 1 ACCS : BOOLEAN;
58: D 1 ENAB : BOOLEAN;
59: D 1 BUSY : BOOLEAN;
60: D 1 END;
```

```
61: D 1 EPTYPE = PACKED RECORD
62: D 1 B0 : BYTE;
63: D 1 B1 : BYTE;
64: D 1 B2 : BYTE;
65: D 1 B3 : BYTE;
66: D 1 CASE BOOLEAN OF
67: D 1 TRUE : (CONTROL: STATREG);
68: D 1 FALSE : (STATUS: STATREG);
69: D 1 END;
70: D 1 EPTR = ^EPTYPE;
71: D 1 INFOPTR = ^EPINFOREC;
72: S
73: S
74: D 1 PROCEDURE EPPROG( SCODE : INTEGER;
75: D 2 OP : EOCTYPE;
76: D 2 ANYVAR BUFFER : WINDOW;
77: D 2 BUFSIZE : INTEGER;
78: D 2 VAR POSIT : INTEGER;
79: D 2 VAR ECODE : EPEROR);
80: D 2 CONST
81: D 2 LO_ROM = HEX('20000'); { 128k }
82: D 2 HI_ROM = HEX('200000'); { 2048k }
83: D 2 STEPSIZE = HEX('20000'); { 128k }
84: D 2 TYPE
85: D 2 TWOBYTES = PACKED ARRAY[0..1] OF CHAR;
86: D 2 BREC = RECORD
87: D 2 CASE INTEGER OF
88: D 2 0 : (INT: INTEGER);
89: D 2 1 : (BPTR: ^BYTE);
90: D 2 2 : (WPTR: ^TWOBYTES);
91: D 2 3 : (CPTR: ^CHAR);
92: D 2 4 : (WINPTR: WINDOWP);
93: D 2 END;
94: S
95: D 2 VAR
96: D 2 -4 CARD : EPTR;
97: D 2 -8 BYTE : BREC;
98: D 2 -12 DUMMY : INTEGER;
99: D 2 -13 DONE : BOOLEAN;
100: D 2 -18 ENOSCAN : INTEGER;
101: D 2 -22 I : INTEGER;
102: D 2 -24 WBYTES : TWOBYTES;
103: D 2 -28 INFO : INFOPTR;
104: S
105: D 2 PROCEDURE BADIO(ECCODE: EPEROR);
106: D 2 BEGIN
107: D 2 ECODE := ECCODE; ESCAPE(0);
108: D 2 ENO;
109: S
110: D 2 PROCEDURE SETUP;
111: D 2 BEGIN
112: D 2 WITH INFO^ DO
113: D 2 BEGIN
114: D 2 IF EPSTART=0 THEN BADIO(ENOEPROM);
115: D 2 ENOSCAN := POSIT + BUFSIZE;
116: D 2 IF (POSIT < EPSTART) OR (ENOSCAN > EPEND) THEN BADIO(EBADARGS);
117: D 2 BYTE.INT := POSIT;
118: D 2 DONE := FALSE;
119: D 2 END;
120: D 2 END;
```



```

121:S
122:D      2  PROCEDURE GETINFO(ANYVAR UNFO : EPINFOREC);
123:C      3  BEGIN
124:C      3    UNFO := INFO^;
125:C      3  END;
126:S
127:C      2  BEGIN ( EPPROG )
128:C      2    ECODE := ENDERROR;
129:C      2    TRY
130:C      2      { VALIDATE THE SELECT CODE }
131:C      3    IF (SCODE<IDMINISC) OR (SCODE>IDMAXISC) THEN BADIO(ENOTPRG);
132:C      3    WITH ISC_TABLE(SCODE) DO
133:C      4    BEGIN
134:C      4      CARD := CARD_PTR;
135:C      4      IF (CARD=NULL) OR (CARD_TYPE=0) THEN BADIO(ENOTPRG);
136:C      4      IF ((CARD_TYPE=1) AND (CARD_ID=0)) OR
137:C      5      ((CARD_TYPE=9) AND (CARD_ID=27)) THEN
138:C      5      BEGIN
139:C      5        IF CARD^.ID<>27 THEN BADIO(ENOTPRG);
140:C      5      END
141:C      5      ELSE BADIO(ENOTPRG);
142:C      4      INFO := ADDR(ID_TMP_PTR^.DRV_MISC);
143:C      4    END;
144:C      2    CASE OP OF
145:C      4      EPNIT: { INITIALIZE THE CONTROL RECD }
146:C      4      BEGIN { VALIDATE THE CARD ID }
147:C      4        WITH ISC_TABLE(SCODE) , INFO^ DO
148:C      5        BEGIN
149:C      5          IF (CARD_TYPE=1) AND (CARD_ID=0) THEN
150:C      5          BEGIN { FIX CARD TYPE AND ID }
151:C      6            CARD_TYPE := 9; CARD_ID := 27;
152:C      6          END;
153:C      5          WITH CARD^ DO
154:C      6          BEGIN
155:C      6            ID := 0; { RESET THE CARD }
156:C      6            { INITIALIZE THE CONTROL RECORD }
157:C      6            BFAST := STATUS.FAST; { SAVE PROGRAMMING SPEED }
158:C      6            EPSTART := 0; { ADDRESS OF ATTACHED EPROM CARD }
159:C      6            EPEND := 0; { ADDRESS OF LAST EPROM BYTE + 1 }
160:C      6            { FINDOUT WHICH EPROM CARD IS ATTACHED TO THE PROGRAMMER CARD }
161:C      6            PBYTE.INT := LO_ROM;
162:C      6            DONE := FALSE;
163:C      6            CACHE_OFF; { 3.0 BUG FIX -- 3/13/84 }
164:C      6            REPEAT
165:C      7            TRY
166:C      7              DUMMY := PBYTE.BPTR^; { ACCESS THE EPROM SPACE }
167:C      7              IF STATUS.ACCS THEN
168:C      8              BEGIN { FOUND THE EPROM CARD }
169:C      8                DONE := TRUE;
170:C      8                EPSTART := PBYTE.INT;
171:C      8                EPEND := EPSTART + STEPSIZE;
172:C      8                PBYTE.INT := EPEND;
173:C      8                { CHECK FOR xx64 PARTS OR xx128 PARTS }
174:C      8                IF EPEND<HI_ROM THEN { DON'T PASS END OF ROM }
175:C      9                BEGIN
176:C      10                CONTROL.ACCS := FALSE; { CLEAR THE ACCESS BIT }
177:C      10                DUMMY := PBYTE.BPTR^; { ACCESS THE EPROM SPACE }
178:C      10                IF STATUS.ACCS THEN EPEND := EPEND + STEPSIZE;
179:C      10                END;
180:C      9            END;

```

```

181:C      9      ELSE
182:C      9        PBYTE.INT := PBYTE.INT + STEPSIZE;
183:C      8      RECOVER { IGNORE BUS ERRORS }
184:C      8      IF ESCAPECODE<-12 THEN ESCAPE(ESCAPECODE)
185:C      8      ELSE
186:C      9        PBYTE.INT := PBYTE.INT + STEPSIZE;
187:C      7        DONE:=DONE OR (PBYTE.INT>HI_ROM);
188:C      7        UNTIL DONE;
189:C      6        CACHE_ON; { 3.0 BUG FIX 3/13/84 }
190:C      6        IF EPSTART=D THEN BADIO(ENDEPROM);
191:C      6      END;
192:C      5      END;
193:C      4      END;
194:S
195:C      4      EPREAD: { MOVE DATA FROM EPROM SPACE TO BUFFER }
196:C      4      IF BUFSIZE>0 THEN
197:C      5      BEGIN
198:C      5        SETUP;
199:C      5        MOVELEFT(PBYTE.CPTR^,BUFFER[0],BUFSIZE);
200:C      5        POSIT := ENDSCAN;
201:C      5      END;
202:S
203:C      4      EPPRITE,EPCWRITE: { MOVE DATA FROM BUFFER TO EPROM }
204:C      4      IF BUFSIZE>0 THEN
205:C      5      BEGIN
206:C      5        SETUP;
207:C      5        WITH CARD^, INFO^ DO
208:C      6        BEGIN
209:C      6          TRY
210:C      7            CONTROL.FAST := BFAST; { SET BURN SPEED }
211:C      7            CONTROL.ENAB := TRUE; { ENABLE WRITE }
212:C      7            I := 0;
213:C      7            IF ODD(PBYTE.INT) THEN
214:C      8            BEGIN { BURN FIRST BYTE }
215:C      8              IF PBYTE.CPTR^<>BUFFER[I] THEN
216:C      9              BEGIN { PATTERN NOT SAME SO BURN IT }
217:C      9                PBYTE.CPTR^ := BUFFER[I];
218:C      9                WHILE CONTROL.BUSY DO; { WAIT FOR BURN TO FINISH }
219:C      9                IF OP = EPCWRITE THEN { CHECK PATTERN AFTER BURN }
220:C      10                IF PBYTE.CPTR^<>BUFFER[I] THEN BADIO(EPFAIL);
221:C      10              END;
222:C      8              I := I + 1;
223:C      8              PBYTE.INT := PBYTE.INT + 1;
224:C      8            END;
225:C      7            ENDSCAN := BUFSIZE-1;
226:C      7            WHILE I<BUFSIZE DO
227:C      8            BEGIN { BURN 2 BYTES AT A TIME EXCEPT FOR LAST ODD BYTE }
228:C      8              IF I=ENDSCAN THEN
229:C      9              BEGIN { BURN LAST BYTE }
230:C      9                IF PBYTE.CPTR^<>BUFFER[I] THEN
231:C      10                BEGIN { PATTERN NOT SAME SO BURN IT }
232:C      10                  PBYTE.CPTR^ := BUFFER[I];
233:C      10                  WHILE CONTROL.BUSY DO; { WAIT FOR BURN TO FINISH }
234:C      10                  IF OP = EPCWRITE THEN { CHECK PATTERN AFTER BURN }
235:C      11                  IF PBYTE.CPTR^<>BUFFER[I] THEN BADIO(EPFAIL);
236:C      11                END;
237:C      9                I := I + 1;
238:C      9                PBYTE.INT := PBYTE.INT + 1;
239:C      9              END
240:C      8            ELSE

```



```

241:C      BEGIN { BURN TWO BYTES }
242:C      WBYTES[0] := BUFFER[I];
243:C      WBYTES[1] := BUFFER[I+1];
244:C      IF PBYTE.WPTR<>WBYTES THEN
245:C      BEGIN { PATTERN NOT SAME SO BURN IT }
246:C      PBYTE.WPTR := WBYTES; { BURN BOTH BYTES }
247:C      WHILE CONTROL.BUSY DO; { WAIT FOR BURN TO FINISH }
248:C      IF OP=EPCWRITE THEN { CHECK PATTERN AFTER BURN }
249:C      IF PBYTE.WPTR<>WBYTES THEN BADIO(EPFAIL);
250:C      END;
251:C      I := I + 2;
252:C      PBYTE.INT := PBYTE.INT + 2;
253:C      END; { BURN TWO BYTES }
254:C      END; { WHILE I<BUFSIZE }
255:C      CONTROL.ENAB:=FALSE; { TURN OFF WRITE ENABLE }
256:C      POSIT := PBYTE.INT; { RECORD THE POSITION }
257:C      RECOVER
258:C      BEGIN
259:C      CONTROL.ENAB:=FALSE; { TURN OFF WRITE ENABLE }
260:C      POSIT := PBYTE.INT; { RECORD THE POSITION }
261:C      IF ESCAPECODE<>0 THEN ESCAPE(ESCAPECODE);
262:C      END;
263:C      END; { WITH CARD^, INFO^ }
264:C      END;
265:S
266:C      4      EPBLANK: { CHECK THE EPROM SPACE FOR HEX FF }
267:C      4      IF BUFSIZE>0 THEN
268:C      5      BEGIN
269:C      5      SETUP;
270:C      5      REPEAT
271:C      6      IF PBYTE.WPTR[0]<>CHR(255) THEN DONE := TRUE
272:C      7      ELSE PBYTE.INT := PBYTE.INT + 1;
273:C      6      DONE := DONE OR (PBYTE.INT>ENDSCAN);
274:C      6      UNTIL DONE;
275:C      6      POSIT := PBYTE.INT;
276:C      6      IF POSIT<ENDSCAN THEN BADIO(ENOTBLANK);
277:C      5      END;
278:S
279:C      4      EPCHECK: { CHECK DATA FOR POSSIBLE BURN }
280:C      4      IF BUFSIZE>0 THEN
281:C      5      BEGIN
282:C      5      SETUP;
283:C      5      I := 0;
284:C      5      REPEAT
285:C      6      IF IOR(OPD(PBYTE.WINPTR[I]),ORD(BUFFER[I]))=ORD(PBYTE.WINPTR[I])
286:C      7      THEN I := I + 1;
287:C      6      ELSE DONE := TRUE;
288:C      6      DONE := DONE OR (I>=BUFSIZE);
289:C      6      UNTIL DONE;
290:C      6      POSIT := PBYTE.INT + I;
291:C      6      IF I<BUFSIZE THEN BADIO(ECFAIL);
292:C      5      END;
293:S
294:C      4      EPBRATE: INFO^.BFAST := POSIT<>0; { SET BURN RATE }
295:S
296:C      4      EPINFO: GETINFO(BUFFER); { GET BURN RATE AND ADDRESS INFO }
297:S
298:C      4      END; { CASE }
299:C      4      RECOVER
300:C      4      BEGIN

```

```

301:C      3      IF ESCAPECODE<>0 THEN ESCAPE(ESCAPECODE);
302:C      3      END;
303:C      2      END; { EPPROG }
304:S
305:S
306:C      1      FUNCTION EPROG(      SCODE : INTEGER;
307:C      2      OP : EOPTYPE;
308:C      2      ANYVAR : BUFFER : WINDOW;
309:C      2      BUFSIZE : INTEGER;
310:C      2      VAR POSIT : INTEGER):EPERROR;
311:C      2      VAR
312:C      2      EOP : EOPTYPE;
313:C      2      ECODE : EPERROR;
314:C      -4      BEGIN
315:C      2      CASE OP OF
316:C      3      EREAD : EOP:=EPREAD;
317:C      3      EWRITE : EOP:=EWRITE;
318:C      3      EPCWRITE : EOP:=EPCWRITE;
319:C      3      EPBLANK : EOP:=EPBLANK;
320:C      3      ECHECK : EOP:=ECHECK;
321:C      2      END;
322:C      2      EPPROG(SCODE,EOP,BUFFER,BUFSIZE,POSIT,ECODE); EPROG:=ECODE;
323:C      2      END;
324:S
325:C      1      FUNCTION EGETINFO(      SCODE : INTEGER;
326:C      2      VAR INFO : EPINFO):EPERROR;
327:C      2      VAR
328:C      2      I : INTEGER;
329:C      -4      ECODE : EPERROR;
330:C      -6      BEGIN
331:C      2      EPPROG(SCODE,EPINFO,INFO,0,I,ECODE); EGETINFO:=ECODE;
332:C      2      END;
333:S
334:C      1      FUNCTION EBRATE(SCODE : INTEGER;BRATE : BOOLEAN):EPERROR;
335:C      2      VAR
336:C      2      I : INTEGER;
337:C      -4      ECODE : EPERROR;
338:C      -6      BEGIN
339:C      2      I := ORD(BRATE);
340:C      2      EPPROG(SCODE,EBRATE,I,0,I,ECODE); EBRATE:=ECODE;
341:C      2      END;
342:S
343:C      1      FUNCTION EINIT(SCODE : INTEGER):EPERROR;
344:C      2      VAR
345:C      2      I : INTEGER;
346:C      -4      ECODE : EPERROR;
347:C      -6      BEGIN
348:C      2      EPPROG(SCODE,EINIT,I,0,I,ECODE); EINIT:=ECODE;
349:C      2      END;
350:S
351:C      1      END.
352:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

EPROMS

Description

EPROMS contains the TM (transfer method) for EPROMs.

Requirements

SYSGLOBALS and ASM.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:D 1: $SYSPROC$
2:D 2: $DEBUG OFF$ $RANGE OFF$
3:D 3: PROGRAM INST EPROM;
4:D 4: MODULE EPROMS;
5:D 5: IMPORT
6:D 6: SYSGLOBALS,ASM;
7:D 7: EXPORT
8:D 8: PROCEDURE EPROM_TM(FP:FIBP; REQUEST:AMREQUESTTYPE;
9:D 9: ANYVAR BUFFER:WINDOW; BUFSIZE,POSITION: INTEGER);
10:D 10: IMPLEMENT
11:S 11:
12:D 12: PROCEDURE EPROM_TM(FP:FIBP; REQUEST:AMREQUESTTYPE;
13:D 13: ANYVAR BUFFER:WINDOW; BUFSIZE,POSITION: INTEGER);
14:D 14: CONST
15:D 15: LO_ROM = HEX('20000');
16:D 16: HI_ROM = HEX('20000');
17:D 17: STEPSIZE = HEX('4000');
18:D 18: ROMHEADER = HEX('FOFF');
19:D 19: HEADERSIZE = 18;
20:D 20: K128 = 131072;
21:D 21: K256 = 262144;
22:D 22: K16 = 16384;
23:D 23: TYPE
24:D 24: HRECORD = PACKED RECORD
25:D 25: HEADER : 0..65535;
26:D 26: IDCHAR : BYTE;
27:D 27: FLAG : BYTE;
28:D 28: END;
29:S 29:
30:D 30: HPTR = ^HRECORD;
31:D 31: SCANREC = RECORD
32:D 32: CASE INTEGER OF
33:D 33: 0 : (INT:INTEGER);
34:D 34: 1 : (PTR:HPTR);
35:D 35: 2 : (WPTR:WINDOWP);
36:D 36: 3 : (CPTR:CHAR);
37:D 37: END;
38:D 38: VAR
39:D 39: SCANPTR : SCANREC;
40:D 40: COUNTER : INTEGER;
41:D 41: DONE : BOOLEAN;
42:D 42: SIZE : INTEGER;
43:S 43:
44:C 44: BEGIN
45:C 45: IORESULT := ORD(INOERROR);
46:C 46: IF (UNITABLE^[FP^.FUNIT].BYTEOFFSET=0) AND
47:C 47: (REQUEST<>CLEARUNIT) THEN IORESULT:=ORD(ZNODEVICE)
48:C 48: ELSE
49:C 49: CASE REQUEST OF
50:C 50: READBYTES: ( MOVE DATA FROM EPROM TO BUFFER )
51:C 51: IF BUFSIZE>0 THEN
52:C 52: WITH FP^ DO
53:C 53: BEGIN
54:C 54: IF (POSITION+BUFSIZE)>FPEOF THEN IORESULT := ORD(ZNOSUCHBLK)
55:C 55: ELSE
56:C 56: BEGIN
57:C 57: COUNTER := HEADERSIZE + FILEID + POSITION;
58:C 58: COUNTER := COUNTER+(2*(COUNTER DIV K16))+
59:C 59: UNITABLE^[FUNIT].BYTEOFFSET;
60:C 60: SCANPTR.INT := COUNTER;

```

```

61:C 61: IF BUFSIZE=1 THEN BUFFER[0] := SCANPTR.CPTR^
62:C 62: ELSE
63:C 63: BEGIN ( MOVING MORE THAN ONE BYTE OF DATA )
64:C 64: COUNTER := 0;
65:C 65: REPEAT
66:C 66: SIZE := (SCANPTR.INT + K16) DIV K16 * K16 - SCANPTR.INT;
67:C 67: IF SIZE>BUFSIZE THEN SIZE := BUFSIZE;
68:C 68: MOVELEFT(SCANPTR.CPTR^,BUFFER[COUNTER],SIZE);
69:C 69: BUFSIZE := BUFSIZE - SIZE;
70:C 70: COUNTER := COUNTER + SIZE;
71:C 71: SCANPTR.INT := SCANPTR.INT + SIZE + 2;
72:C 72: UNTIL BUFSIZE=0;
73:C 73: END;
74:C 74: END;
75:C 75: { READBYTES }
76:C 76: WRITEBYTES: IORESULT := ORD(ZPROTECTED);
77:C 77: FLUSH;
78:C 78: CLEARUNIT:
79:C 79: BEGIN ( FIND THE NTH DISC HEADER )
80:C 80: SCANPTR.INT := LO_ROM;
81:C 81: UNITABLE^[FP^.FUNIT].BYTEOFFSET := 0; ( CLEAR THE OFFSET )
82:C 82: COUNTER := UNITABLE^[FP^.FUNIT].DV;
83:C 83: DONE := FALSE;
84:C 84: REPEAT ( FIND THE EPROM DISC HEADER )
85:C 85: TRY
86:C 86: IF SCANPTR.PTR^.HEADER=ROMHEADER THEN
87:C 87: BEGIN
88:C 88: IF (SCANPTR.PTR^.FLAG=HEX('0B')) OR
89:C 89: (SCANPTR.PTR^.FLAG=HEX('1B')) THEN
90:C 90: BEGIN ( FOUND DISC HEADER )
91:C 91: COUNTER := COUNTER - 1; ( COUNT IT )
92:C 92: DONE := COUNTER<0; ( IS THIS THE ONE )
93:C 93: END;
94:C 94: END;
95:C 95: IF NOT DONE THEN SCANPTR.INT := SCANPTR.INT + STEPSIZE;
96:C 96: RECOVER ( IGNORE BUS ERRORS )
97:C 97: IF ESCAPECODE<>~12 THEN ESCAPE(ESCAPECODE);
98:C 98: ELSE SCANPTR.INT := SCANPTR.INT + STEPSIZE;
99:S 99:
100:C 100: DONE := DONE OR (SCANPTR.INT>=HI_ROM);
101:C 101: UNTIL DONE;
102:C 102: IF SCANPTR.INT>=HI_ROM THEN IORESULT := ORD(ZNODEVICE)
103:C 103: ELSE
104:C 104: WITH UNITABLE^[FP^.FUNIT] DO
105:C 105: BEGIN
106:C 106: BYTEOFFSET := SCANPTR.INT;
107:C 107: UMAXBYTES := (SCANPTR.INT+K256) DIV K256 * K256 - SCANPTR.INT;
108:C 108: END;
109:C 109: END; ( CLEARUNIT )
110:C 110: OTHERWISE
111:C 111: IORESULT := ORD(ZBADMODE);
112:C 112: END; ( CASE REQUEST )
113:C 113: END;
114:C 114: ENO; ( MODULE EPROM_MODULE )
115:S 115:
116:D 116: ( EPROMS INSTALLATION PROGRAM )
117:D 117: IMPORT LOADER;
118:C 118: BEGIN
119:C 119: MARKUSER;
120:C 120: END.

```


No errors. No warnings.

***** Nonstandard language features enabled *****

Description

ETU contains an interactive program for controlling the EPROM programming card.

Requirements

SYSGLOBALS, MISC, ASM, FS, CI, LOADER, SYSDEVS, LDR, and EDRIIVER.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1: S (*
2: S
3: S
4: S (c) Copyright Hewlett-Packard Company, 1983.
5: S All rights are reserved. Copying or other
6: S reproduction of this program except for archival
7: S purposes is prohibited without the prior
8: S written consent of Hewlett-Packard Company.
9: S
10: S
11: S RESTRICTED RIGHTS LEGEND
12: S
13: S Use, duplication, or disclosure by the Government
14: S is subject to restrictions as set forth in
15: S paragraph (b) (3) (8) of the Rights in Technical
16: S Data and Computer Software clause in
17: S FAR 7-104.9(a).
18: S
19: S HEWLETT-PACKARD COMPANY
20: S 0 Fort Collins, Colorado *)
21: S
22: S 0 $COPYRIGHT 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'S
23: S 0 $SYSPROG$ $DEBUG OFF$ $RANGE OFF$
24: S
25: S 0 PROGRAM ETU(KEYBOARD,INPUT,OUTPUT);
26: S 1 $SEARCH 'LI8R:EDRIVER'$
27: S 1 IMPORT SYSGL08ALS,MISC,FS,CI,EDRIVER,ASM,SYSDEVS;
28: S
29: S 1 VAR
30: S 1 KEYBOARD TEXT;
31: S
32: S 1 (*****
33: S 2 PROCEDURE COMMANDLEVEL;
34: S 2 CONST
35: S 2 SH_EXC = CHR(27);
36: S 2 K16 = 16384;
37: S 2 MINS = 8;
38: S 2 MAXSC = 31;
39: S
40: S 2 TYPE
41: S 2 PROMPTTYPE = STRING80;
42: S 2 BUFTYPE = PACKED ARRAY[0..MAXINT] OF CHAR;
43: S 2 BIGPTR = ^BUFTYPE;
44: S 2 PASSTYPE = (CHECK,BURN,VERIFY);
45: S 2 TYPE
46: S 2 EPROMPTR = ^EPROMREC;
47: S 2 EPROMREC = RECORD
48: S 2 NEXT : EPROMPTR;
49: S 2 BASEADDR : INTEGER;
50: S 2 EPSIZE : INTEGER;
51: S 2 EPINC : INTEGER;
52: S 2 PRESENT : ARRAY[0..7] OF BOOLEAN;
53: S 2 ADDRESS : ARRAY[0..7] OF INTEGER;
54: S 2 END;
55: S 2 DIRSTATUS = (DNEEDED,DWANTED,DONTCARE);
56: S 2 CONTROL = RECORD
57: S 2 CF18 : F18;
58: S 2 PATH : INTEGER;
59: S 2 DIROPEN : BOOLEAN;
60: S 2 FILEOPEN : BOOLEAN;
61: S 2 USEUNIT : BOOLEAN;

```

```

61: D 2 MOUNTED : BOOLEAN;
62: D 2 CPVOL : VIO;
63: D 2 CVOL : VIO;
64: D 2 CFILE : FID;
65: D 2 DSTATUS : DIRSTATUS;
66: D 2 END;
67: S
68: S 2 VAR
69: S -4 SCODE : INTEGER;
70: S -5 OP : CHAR;
71: S -6 FASTBURN : BOOLEAN;
72: S -10 TEMP : INTEGER;
73: S -20 EPINFO : EPINFOREC;
74: S -22 ERROR : EPERROR;
75: S
76: S -23 2 HEAPINUSE : BOOLEAN;
77: S
78: S -854 2 ININFO : CONTROL;
79: S
80: S -858 2 SAVEIO : INTEGER;
81: S -862 2 SAVEESC : INTEGER;
82: S -866 2 LHEAP : ANYPTR;
83: S -870 2 EPROMLIST : EPROMPTR;
84: S -874 2 EPROMDATA : EPROMPTR;
85: S -878 2 LEFTTOXFER : INTEGER;
86: S -882 2 OUTPOSITION : INTEGER;
87: S -886 2 OUTSTARTA : INTEGER;
88: S -890 2 PASS : INTEGER;
89: S
90: S -890 2 (*****
91: S 2 PROCEDURE FIXLOCK;
92: S 3 BEGIN
93: S 4 IF LOCKLEVEL<>0 THEN
94: S 5 BEGIN LOCKLEVEL := 1; LOCKDOWN; END;
95: S 3 END; { FIXLOCK }
96: S
97: S -890 2 (*****
98: S 2 PROCEDURE PRINTIOERRMSG;
99: S 3 VAR
100: S -82 MSG : STRING[80];
101: S 3 BEGIN
102: S 4 IF IORESULT<>ORD(INOERROR) THEN
103: S 5 BEGIN
104: S 6 GETIOERRMSG(MSG,IORESULT);
105: S 6 WRITELN('Error: ',MSG,CTEOL);
106: S 6 IF STREAMING THEN ESCAPE(-1);
107: S 5 END;
108: S 3 END; { PRINTIOERRMSG }
109: S
110: S -890 2 (*****
111: S -82 2 PROCEDURE SHOWPROMPT(P : PROMPTTYPE);
112: S 3 BEGIN WRITE(HOMECHAR,P,CTEDL); END;
113: S
114: S -890 2 (*****
115: S 2 PROCEDURE GOODIO;
116: S 3 BEGIN IF IORESULT<>ORD(INOERROR) THEN ESCAPE(0); END;
117: S
118: S -890 2 (*****
119: S 2 PROCEDURE BADIO(IOC00E : IORESULT);
120: S 3 BEGIN IORESULT := ORD(IOC00E), ESCAPE(0); END;

```



```

121:S
122:D -890 2 (*****
123:D -82 2 PROCEDURE BADMESSAGE(P : PROMPTTYPE);
124:C 3 BEGIN
125:C 3   WRITELN(P,CTEOL);
126:C 3   IF STREAMING THEN ESCAPE(-1) ELSE BADIO(INOERROR);
127:C 3 END; { BADMESSAGE }
128:S
129:D -890 2 (*****
130:D 2 PROCEDURE BADCOMMAND(C:CHAR);
131:C 3 BEGIN
132:C 3   WRITELN('BAD COMMAND ',C,',',CTEOL);
133:C 3   IF STREAMING THEN ESCAPE(-1) ELSE BADIO(INOERROR);
134:C 3 END; { BADCOMMAND }
135:S
136:D -890 2 (*****
137:D 2 PROCEDURE READCHECK;
138:C 3 BEGIN
139:C 3   IF IORESULT<>ORD(INOERROR) THEN
140:C 4   BEGIN
141:C 4     SAVEIO := IORESULT; WRITELN; IORESULT := SAVEIO;
142:C 4     ESCAPE(0);
143:C 4   END;
144:C 3 END; { READCHECK }
145:S
146:D -890 2 (*****
147:D 2 PROCEDURE READNUMBER(VAR INT : INTEGER);
148:D 3 VAR
149:D -4 3   I : INTEGER;
150:D -8 3   TI : INTEGER;
151:D -30 3   INSTRING : STRING[20];
152:C 3 BEGIN
153:C 3   READLN(INSTRING); GOODIO;
154:C 3   INSTRING:=STRLTRIM(INSTRING);
155:C 3   IF STRLEN(INSTRING)>0 THEN
156:C 4   TRY
157:C 5   IF INSTRING(1)=SH_EXC THEN ESCAPE(0);
158:C 5   ZAPSPACES(INSTRING);
159:C 5   IF STRLEN(INSTRING)>0 THEN
160:C 6   BEGIN
161:C 6     TI := 0;
162:C 6     FOR I:=1 TO STRLEN(INSTRING) DO
163:C 7     IF (INSTRING[I]<'0') OR (INSTRING[I]>'9') THEN BADIO(IBADVALUE)
164:C 8     ELSE TI := TI * 10 + (ORD(INSTRING[I]) - ORD('0'));
165:C 8     INT := TI;
166:C 6   END;
167:C 6   RECOVER
168:C 5   IF ESCAPECODE=-4 THEN BADIO(IBADVALUE)
169:C 5   ELSE ESCAPE(ESCAPECODE);
170:C 3 END; { READNUMBER }
171:S
172:D -890 2 (*****
173:D 2 FUNCTION UNITNUMBER(VAR FVID : VID):BOOLEAN;
174:D 3 LABEL 1;
175:D 3 VAR
176:D -6 3   SL,I : INTEGER;
177:C 3 BEGIN
178:C 3   UNITNUMBER := FALSE;
179:C 3   SL := STRLEN(FVID);
180:C 3   IF SL<2 THEN GOTO 1;

```

```

181:C 3   IF FVID[1]<>'#' THEN GOTO 1;
182:C 3   FOR I:=2 TO SL DO IF (FVID[I]<'0') OR (FVID[I]>'9') THEN GOTO 1;
183:C 3   UNITNUMBER := TRUE;
184:C 3   1:END;
185:D -890 2 { UNITNUMBER }
186:D -890 2 (*****
187:D 2 PROCEDURE UPCCHAR(VAR CH : CHAR);
188:C 3 BEGIN IF ('a'<=CH) AND (CH<='z') THEN CH:=CHR(ORD(CH)-32); END;
189:S
190:D -890 2 (*****
191:D 2 PROCEDURE PROMPTREAD(P:PROMPTTYPE; VAR ANSWER:CHAR; LIST:PROMPTTYPE;
192:D -164 3   DEFAULT:CHAR);
193:D -164 3 LABEL 1;
194:D -164 3 VAR
195:D -168 3   I : INTEGER;
196:C 3 BEGIN
197:C 3   IF (DEFAULT<>SH_EXC) AND STREAMING THEN ANSWER:=DEFAULT
198:C 4   ELSE
199:C 4   BEGIN
200:C 4     WRITE(P,CTEOL);
201:C 4     REPEAT
202:C 5     READ(KEYBOARD, ANSWER); READCHECK; UPCCHAR(ANSWER);
203:C 5     IF ANSWER=SH_EXC THEN BEGIN WRITELN; BADIO(INOERROR); END;
204:C 5     FOR I:=1 TO STRLEN(LIST) DO { IS CHARACTER IN THE LIST ? }
205:C 6     IF ANSWER=LIST[I] THEN GOTO 1;
206:C 5     IF STREAMING THEN BADCOMMAND(ANSWER);
207:C 5     UNTIL FALSE;
208:C 4     1:WRITELN(ANSWER);
209:C 4   END;
210:C 3 END; { PROMPTREAD }
211:S
212:D -890 2 (*****
213:D -82 2 PROCEDURE PROMPTYORN(P : PROMPTTYPE; VAR ANSWER :CHAR);
214:C 3 BEGIN
215:C 3   PROMPTREAD(P+' ? (Y/N) ',ANSWER,'YN','Y');
216:C 3 END; { PROMPTYORN }
217:S
218:D -890 2 (*****
219:D -82 2 PROCEDURE MOUNTVOLUME(SD : PROMPTTYPE ,VAR FINFO : CONTROL);
220:D 3 VAR
221:D -83 3   ANSWER : CHAR;
222:D -88 3   UNIT : INTEGER;
223:D -106 3   TEMPNAME : VID;
224:S
225:C 3 BEGIN
226:C 3   WITH FINFO DO
227:C 4   BEGIN
228:C 4     IF STREAMING THEN
229:C 5     BEGIN
230:C 5       WRITELN('VOLUME ',CPVOL,' NOT ONLINE WHILE STREAMING',CTEOL);
231:C 5       ESCAPE(-1);
232:C 5     END;
233:S
234:C 4   TEMPNAME := CPVOL;
235:C 4   UNIT := FINDVOLUME(TEMPNAME,FALSE); { CHECK FOR BAD UNIT # }
236:C 4   IORESULT := DRD(INOERROR);
237:S
238:C 4   REPEAT
239:C 5   { CONSTRUCT THE PROMPT }
240:C 5   WRITE('Please mount',SD);

```



```

241:C      5      IF STRLEN(CVOL)>0 THEN WRITE(' volume ',CVOL);
242:C      5      IF ((STRLEN(SD)>0) OR (STRLEN(CVOL)>0)) AND USEUNIT THEN WRITE(' in');
243:C      5      IF USEUNIT THEN WRITE(' unit ',CPVOL);
244:C      5      WRITELN(CTEOL);
245:C      5      PROMPTREAD('C' continues, <sh_exc> aborts ',ANSWER,'C','C');
246:C      5      IF USEUNIT THEN TEMPNAME := CPVOL ELSE TEMPNAME := CVOL;
247:C      5      CFIB.FUNIT := FINDVOLUME(TEMPNAME,TRUE);
248:C      5
249:C      5
250:C      5      IF CFIB.FUNIT>0 THEN
251:C      6      BEGIN
252:C      6      IF IORESULT=ORD(INODIRECTORY) THEN
253:C      7      BEGIN
254:C      7      IF DSTATUS<>DONTCARE THEN WRITELN('NO DIRECTORY ON ',CPVOL);
255:C      7      SETSTRLEN(TEMPNAME,0);
256:C      7      CASE DSTATUS OF
257:C      8      DNEEDED: CFIB.FUNIT := 0;
258:C      8      DWANTED: BEGIN
259:C      8      PROMPTYORN('Use current media',ANSWER);
260:C      8      IF ANSWER='N' THEN CFIB.FUNIT := 0
261:C      8      ELSE DSTATUS := DONTCARE;
262:C      8      END;
263:C      8      OTHERWISE
264:C      8      END; { CASE DSTATUS }
265:C      7      END;
266:C      7      ELSE
267:C      7      BEGIN
268:C      7      IF IORESULT<>ORD(INOERROR) THEN
269:C      8      BEGIN
270:C      8      PRINTIOERRMSG; CFIB.FUNIT := 0;
271:C      8      END
272:C      8      ELSE
273:C      8      BEGIN { FOUND A DIRECTORY }
274:C      8      IF CVOL='' THEN CVOL := TEMPNAME
275:C      9      ELSE
276:C      9      IF CVOL<>TEMPNAME THEN CFIB.FUNIT := 0;
277:C      8      END;
278:C      7      END;
279:C      6      END;
280:C      5      UNTIL CFIB.FUNIT>0;
281:C      4      CFIB.FVID := CVOL;
282:C      4      MOUNTED := TRUE;
283:C      4      END;
284:C      3      END; { MOUNT VOLUME }
285:C
286:C      -890 2 {*****}
287:C      2      PROCEDURE SPACEWAIT;
288:C      3      VAR
289:C      3      ANSWER : CHAR;
290:C      3      BEGIN
291:C      3      PROMPTREAD(' <space> continues, <sh_exc> aborts ',ANSWER,' ',' ');
292:C      3      END; { SPACEWAIT }
293:C
294:C      -890 2 {*****}
295:C      -82 2      PROCEDURE PROMPTFORCHAR(PL : PROMPTTYPE; VAR CH : CHAR);
296:C      3      BEGIN
297:C      3      FGOTOXY(OUTPUT,D,0); WRITE(PL,' ? ',CTEOL);
298:C      3      READ(KEYBOARD,CH); READCHECK;
299:C      3      UPCHAR(CH); WRITELN(CH);
300:C      3      FGOTOXY(OUTPUT,0,1); WRITELN(CTEOL);

```

```

301:C      3      END; { PROMPTFORCHAR }
302:C
303:C      -890 2 {IOCHECK OFF$}
304:C      -890 2 {*****}
305:C      2      PROCEDURE SETUPFIBFORFILE(FILENAME : FID;
306:C      3      VAR LFIB : FIB;
307:C      3      VAR VNAME : VID);
308:C      -122 3      VAR
309:C      -122 3      LKIND : FILEKIND;
310:C      -128 3      SEGS : INTEGER;
311:C
312:C      3      BEGIN
313:C      3      SEGS := 0;
314:C      3      IORESULT := ORD(INOERROR);
315:C      3      WITH LFIB DO
316:C      4      IF SCANTITLE(FILENAME,FVID,FTITLE,SEGS,LKIND) THEN
317:C      5      BEGIN
318:C      5      VNAME := FVID;
319:C      5      FUNIT := FINDVOLUME(FVID,TRUE);
320:C      5      FKIND := LKIND; FEFT := EFTTABLE^[LKIND];
321:C      5      FOSTRING := NIL; FBUFFERED := TRUE;
322:C      5      FPOS := SEGS * S12; FREPTCNT := 0;
323:C      5      FANONYMOUS := FALSE; FMODIFIED := FALSE;
324:C      5      FBUFFCHANGED := FALSE; FSTARTADDRESS := 0;
325:C      5      FLASTPOS := -1; PATHID := -1;
326:C      5      FVOSRHTMP := TRUE; FLOCKED := TRUE;
327:C      5      FEOF := FALSE; FEOLN := FALSE;
328:C      5      END
329:C      5      ELSE BADIO(ISOFTITLE);
330:C      3      END; { SETUPFIBFORFILE }
331:C
332:C      -890 2 {*****}
333:C      2      PROCEDURE CLOSEDIR;
334:C      3      BEGIN
335:C      3      WITH ININFO, CFIB DO
336:C      4      BEGIN
337:C      4      IF DIROPEN THEN
338:C      5      BEGIN
339:C      5      LOCKUP; { LOCK KEYBOARD FOR THIS OPERATION }
340:C      5      PATHID := PATH; { RESTORE PATHID }
341:C      5      CALL(UNITABLE^[FUNIT].DAM,CFIB,FUNIT,CLOSEDIRECTORY);
342:C      5      DIROPEN := FALSE;
343:C      5      LOCKDOWN;
344:C      5      END;
345:C      4      END;
346:C      3      END; { CLOSEDIR }
347:C
348:C      -890 2 {*****}
349:C      2      PROCEDURE OPENDIR(FILENAME : FID;
350:C      3      VAR SOURCEFILE : FID;
351:C      3      PROMPT : PROMPTTYPE;
352:C      3      VAR FINFO : CONTROL;
353:C      3      VAR DIRCATENTRY : CATENTRY);
354:C      -204 3      VAR
355:C      -204 3      UNIT : INTEGER;
356:C      -208 3
357:C      3      BEGIN { OPENDIR }
358:C      3      IORESULT := ORD(INOERROR);
359:C      3      WITH FINFO, CFIB DO
360:C      4      TRY

```



```

361:C      5      SETUPFIBFORFILE(FILENAME,CFIB,CPVOL);
362:C      5      USEUNIT := UNITNUMBER(CPVOL);  OSTATUS := DNEEDED;
363:C      5      IF USEUNIT THEN CVOL := ' ' ELSE CVOL := CPVOL;
364:C      5      IF (FUNIT=0) OR UNITNUMBER(FVID) THEN MOUNTVOLUME(PROMPT,FINFO)
365:C      6      ELSE MOUNTED := TRUE;
366:C      5      WITH UNITABLE^[FUNIT] DO
367:C      6      BEGIN
368:C      7      LOCKUP; { LOCK KEYBOARD }
369:C      7      FWINDOW := ADDR(DIRCATENTRY);
370:C      7      CALL(DAM,CFIB,FUNIT,OPENDIRECTORY);
371:C      7      DIROPEN := (IORESULT=ORD(INOERRR));
372:C      7      IF DIROPEN THEN
373:C      8      BEGIN
374:C      9      PATH := PATHID;
375:C      9      SOURCEFILE := FTITLE;
376:C      9      CVOL := DIRCATENTRY.CNAME;
377:C      9      END;
378:C      7      LOCKDOWN; { UNLOCK KEYBOARD }
379:C      7      IF NOT DIROPEN THEN ESCAPE(0); { OPENDIRECTORY FAILED }
380:C      6      END
381:C      6      RECOVER
382:C      5      IF ESCAPECODE<>0 THEN ESCAPE(ESCAPECODE);
383:C      3      END; { OPENDIR }
384:C
385:C      -890  2  (*****
386:C      2  PROCEDURE INMOUNT(SWAP : BOOLEAN);
387:C      3  BEGIN
388:C      3  IF NOT ININFO.MOUNTED THEN
389:C      4  WITH ININFO, CFIB DO
390:C      5  BEGIN
391:C      5  MOUNTVOLUME(' SOURCE',ININFO);
392:C      5  UNITABLE^[FUNIT].UMEDIAVALID := TRUE;
393:C      5  END;
394:C      3  END; { INMOUNT }
395:C
396:C      -890  2  (*****
397:C      2  PROCEDURE CLOSEINFILE;
398:C      3  BEGIN
399:C      3  WITH ININFO,CFIB DO
400:C      4  BEGIN
401:C      5  IF FILEOPEN THEN
402:C      6  BEGIN
403:C      7  LOCKUP;
404:C      7  FMODIFIED := FALSE;
405:C      7  CALL(UNITABLE^[FUNIT].DAM,CFIB,FUNIT,CLOSEFILE);
406:C      7  FILEOPEN := FALSE;
407:C      7  LOCKDOWN;
408:C      7  END;
409:C      4  END;
410:C      3  END; { CLOSEINFILE }
411:C
412:C
413:C      -890  2  (*****
414:C      2  PROCEDURE CLOSEALL;
415:C      3  BEGIN CLOSEINFILE; CLOSEDIR; END;
416:C
417:C      -890  2  (*****
418:C      2  PROCEDURE ANYTOMEM( FIB : FIB;
419:C      3  ANYVAR BUFFER : BIGPTR;
420:C      3  MAXBUF : INTEGER);

```

```

421:D      3  VAR
422:D      -4  3  BUFREC : ^STRING255;
423:D      -8  3  BUFPTR : ^CHAR;
424:D      -12 3  LEFTINBUF : INTEGER;
425:D
426:C      3  BEGIN { ANYTOMEM }
427:C      3  BUFPTR := ADDR(BUFFER^);
428:C      3  BUFPTR^ := CHR(0); { DATA COMING }
429:C      3  BUFREC := ADDR(BUFPTR^,1);
430:C      3  SETSTLEN(BUFREC^,0); { ZERO LENGTH RECORD }
431:C      3  BUFPTR := ADDR(BUFREC^,1);
432:C      3  LEFTINBUF := MAXBUF;
433:C
434:C      3  WITH FFIB^, UNITABLE^[FUNIT] DO
435:C      4  BEGIN
436:C      4  CALL(AM,FFIB,READTOEOL,BUFREC^,255,FPOS);
437:C      4  GOODIO;
438:C      4  REPEAT
439:C      5  GOODIO; { CHECK IORESULT FROM LAST READTOEOL }
440:C      5  LEFTTOXFER := LEFTTOXFER - STLEN(BUFREC^);
441:C      5  IF LEFTTOXFER<0 THEN
442:C      6  BEGIN { CLIP THIS LINE AND MAKE END OF FILE }
443:C      6  SETSTLEN(BUFREC^,STLEN(BUFREC^)+LEFTTOXFER);
444:C      6  LEFTTOXFER := 0;
445:C      6  END;
446:C      5  BUFPTR := ADDR(BUFPTR^,STLEN(BUFREC^));
447:C
448:C      5  LEFTINBUF := LEFTINBUF - STLEN(BUFREC^) - 2;
449:C      5  IF STLEN(BUFREC^)=255 THEN BUFPTR := ADDR(BUFPTR^,-1)
450:C      6  ELSE
451:C      6  BEGIN
452:C      6  IF STLEN(BUFREC^)=0 THEN
453:C      7  BEGIN { DISCARD THE LENGTH BYTE }
454:C      7  BUFPTR := ADDR(BUFREC^,-1); LEFTINBUF := LEFTINBUF + 1;
455:C      7  END;
456:C      6  END;
457:C      6  { CHECK END OF LINE/FILE }
458:C      6  CALL(AM,FFIB,READBYTES,BUFPTR^,1,FPOS);
459:C      6  IF FEOLN THEN
460:C      7  BEGIN { END OF LINE }
461:C      7  BUFPTR^ := CHR(1); FEOLN := FALSE;
462:C      7  IF IORESULT = ORD(IEOF) THEN BUFPTR := ADDR(BUFPTR^,1);
463:C      7  END;
464:C      6  IF (IORESULT=ORD(IEOF)) OR (LEFTTOXFER=0) THEN
465:C      7  BEGIN { END OF FILE }
466:C      7  BUFPTR^ := CHR(2);
467:C      7  IORESULT := ORD(INOERRR);
468:C      7  FEOL := TRUE;
469:C      7  END;
470:C      6  GOODIO; { CHECK IORESULT FROM READBYTES }
471:C      6  END;
472:C      5  IF NOT ((LEFTINBUF < 259) OR FEOL) THEN
473:C      6  BEGIN { SETUP FOR TO READ THE NEXT LINE }
474:C      6  BUFPTR := ADDR(BUFPTR^,1);
475:C      6  BUFPTR^ := CHR(0); { DATA RECORD }
476:C      6  BUFREC := ADDR(BUFPTR^,1);
477:C      6  SETSTLEN(BUFREC^,0); { ZERO LENGTH RECORD }
478:C      6  BUFPTR := ADDR(BUFREC^,1);
479:C      6  CALL(AM,FFIB,READTOEOL,BUFREC^,255,FPOS);
480:C      6  END;

```



```

481 C      5      UNTIL (LEFTINBUF < 259) OR FE0F;
482 C      4      BUFPTR := ADDR(BUFPTR,1);    BUFPTR := CHR(3); { END BUFFER }
483 C      4      END;
484 C      3      END; { ANYTOMEM }
485 S
486 D -B90 2 {*****}
487 D 2 FUNCTION CHECKSPACE(START,SIZE:INTEGER; VAR OKSIZE:INTEGER):BOOLEAN;
488 D
489 D      3      VAR
490 D      -4      3      POINT : INTEGER;
491 D      -8      3      PART : INTEGER;
492 D
493 D      3      BEGIN { CHECK SPACE }
494 D      3      OKSIZE := -1; CHECKSPACE := FALSE;
495 D      3      WITH EPROMDATA^ DD
496 D      4      BEGIN
497 D      4      { FIND THE PART CONTAINING THE START ADDRESS }
498 D      4      PART := 0;
499 D      4      WHILE PART < B DD
500 D      5      BEGIN
501 D      5      IF (START >= ADDRESS[PART]) AND
502 D      6      (START < (ADDRESS[PART] + EPINC)) THEN
503 D      6      BEGIN
504 D      6      IF PRESENT[PART] THEN
505 D      7      BEGIN { FOUND START NOW FIND END POINT }
506 D      7      POINT := START + SIZE - 1; { END POINT }
507 D      7      WHILE PART < B DD
508 D      8      BEGIN
509 D      8      IF PRESENT[PART] THEN
510 D      9      BEGIN
511 D      9      IF (POINT < (ADDRESS[PART] + EPINC)) THEN
512 D      10     BEGIN { IT ALL FITS }
513 D      10     OKSIZE := SIZE; CHECKSPACE := TRUE; PART := B; { FORCE EXIT }
514 D      10     END
515 D      10     ELSE { FIGURE SPACE SO FAR }
516 D      10     BEGIN
517 D      10     OKSIZE := (ADDRESS[PART] + EPINC) - START;
518 D      10     PART := PART + 1;
519 D      10     END;
520 D      9     END { IF PRESENT }
521 D      9     ELSE PART := B; { FORCE EXIT }
522 D      8     END; { WHILE }
523 D      7     END; { IF PRESENT }
524 D      6     PART := B; { FORCE EXIT }
525 D      6     END { IF ADDRESS IN RANGE }
526 D      6     ELSE PART := PART + 1;
527 D      4     END; { WHILE }
528 D      4     END; { WITH EPROMDATA }
529 D      3     END; { CHECKSPACE }
530 D
531 D -B90 2 {*****}
532 D 2 PROCEDURE PASSFAILED(FAILCODE:PASSTYPE);
533 D
534 D      3      VAR
535 D      -4      3      PART : INTEGER;
536 D      -5      3      UL : CHAR;
537 D      -10     3      I : INTEGER;
538 D
539 D      3      BEGIN
540 D      3      WITH EPROMDATA^ DD
541 D      4      BEGIN

```

```

541 C      4      FOR I := D TO 7 DO
542 C      5      IF (OUTPOSITION >= ADDRESS[I]) AND
543 C      6      (OUTPOSITION < (ADDRESS[I] + EPINC)) THEN PART := I;
544 C
545 C      4      IF DDD(OUTPOSITION) THEN UL := 'L' ELSE UL := 'U';
546 C      4      WRITE(FAILCODE, ' FAIL');
547 C      4      Writeln(' AT ABSOLUTE ADDRESS ', OUTPOSITION:1);
548 C      4      Writeln(' BYTE POSITION ', OUTPOSITION-OUTSTARTA:1, ' FROM START LOCATION');
549 C      4      Writeln(' EPROM SOCKET ', PART:1, UL
550 C      4      ' BYTE ', (OUTPOSITION-ADDRESS[PART]) DIV 2:1);
551 C      4      END;
552 C      3      IF STREAMING THEN ESCAPE(-1) ELSE BADIO(INOERROR);
553 C      3      END; { PASSFAILED }
554 S
555 D -B90 2 {*****}
556 D 2 PROCEDURE PRINTBR;
557 D
558 D      3      BEGIN
559 D      3      FGOXY(OUTPUT,0,4); WRITE('Burn rate ');
560 D      3      IF FASTBURN THEN WRITE('FAST') ELSE WRITE('SLOW');
561 D      3      Writeln(CTEOL);
562 D
563 D -B90 2 {*****}
564 D 2 PROCEDURE BURNIT(ANYVAR BUFFER : WINDOW;
565 D      3      SIZE : INTEGER);
566 D
567 D      3      VAR
568 D      -4      3      OLDPSIT : INTEGER;
569 D      -12     3      OLDX,OLDY : INTEGER;
570 D
571 D      3      BEGIN
572 D      3      IF PASS=1 THEN
573 D      4      BEGIN { CHECK IF CAN BURN }
574 D      4      ERROR:=EPROG(SCDDE,ECHECK,BUFFER,SIZE,OUTPOSITION);
575 D      4      IF ERROR<>ENOERROR THEN PASSFAILED(CHECK);
576 D      4      END
577 D      4      ELSE
578 D      4      BEGIN { TRY TO BURN IT }
579 D      4      OLDPSIT := OUTPOSITION; { SAVE POSITION FOR RETRY }
580 D      4      ERROR:=EPROG(SCODE,ECURITE,BUFFER,SIZE,OUTPOSITION);
581 D      4      IF ERROR<>ENOERROR THEN
582 D      5      IF NOT FASTBURN THEN
583 D      6      BEGIN
584 D      6      IF ERROR=ECFAIL THEN PASSFAILED(VERIFY)
585 D      6      ELSE PASSFAILED(BURN);
586 D      6      END
587 D      6      ELSE
588 D      6      BEGIN
589 D      6      FASTBURN := FALSE; { SET BURN RATE TO SLOW }
590 D      6      EGETXY(OUTPUT,OLDX,OLDY); { UPDATE THE DISPLAY }
591 D      6      PRINTBR;
592 D      6      FGOXY(OUTPUT,OLDX,OLDY);
593 D      6      ERROR := EBRATE(SCODE,FASTBURN);
594 D      6      OUTPOSITION := OLDPSIT; { RESET POSITION AND RETRY }
595 D      6      ERROR:=EPRDG(SCODE,ECURITE,BUFFER,SIZE,OUTPOSITION);
596 D      6      IF ERROR<>ENOERROR THEN
597 D      7      IF ERROR=ECFAIL THEN PASSFAILED(VERIFY)
598 D      7      ELSE PASSFAILED(BURN);
599 D      6      END;
600 D      3      END;
601 D      3      IORRESULT:=ORD(INOERROR); { CLEAR I/O RESULT }
602 D      3      END; { BURNIT }

```



```

601:S
602:D -890 2 (*****
603:O 2 PROCEDURE MEMTOEPROM(ANYVAR BUFFER : BIGPTR);
604:D 3 VAR
605:D -4 3 BYTES : INTEGER;
606:D -8 3 BUFPTR : ^CHAR;
607:S
608:C 3 BEGIN
609:C 3 BUFPTR := ADDR(BUFFER^);
610:C 3 BEGIN
611:C 3 BYTES := 0;
612:C 3 REPEAT
613:C 4 BUFPTR := ADDR(BUFPTR^,BYTES);
614:C 4 BYTES := ORD(BUFPTR^);
615:C 4 BUFPTR := ADDR(BUFPTR^,1);
616:C 4 CASE 8BYTES OF
617:C 0: BEGIN
618:C 8BYTES := ORD(BUFPTR^); { RECORD LENGTH }
619:C BUFPTR := ADDR(BUFPTR^,1);
620:C 8URNIT(BUFPTR^,8BYTES);
621:C END;
622:C 1: BEGIN { END RECORD }
623:C 8BYTES := 0;
624:C END;
625:C 2: BEGIN { END FILE }
626:C 8BYTES := -1;
627:C END;
628:C 3: BYTES := -1; { END BUFFER }
629:C OTHERWISE IORESULT := ORD(IREADREQUEST);
630:C END;
631:C 4 GOODIO;
632:C 3 UNTIL 8BYTES<0;
633:C 3 END;
634:C 3 END; { MEMTOEPROM }
635:S
636:D -890 2 (*****
637:D 2 FUNCTION CHECKCARD(SC:INTEGER):BOOLEAN;
638:D 3 VAR
639:D -2 3 ECODE : EERROR;
640:C 3 BEGIN
641:C ECODE:=EINIT(SC);
642:C CHECKCARD:=(ECODE=ENCERROR) OR (ECODE=ENOEPRM);
643:C 3 END; { CHECKCARD }
644:S
645:D -890 2 (*****
646:D 2 PROCEDURE CHECKSCODE;
647:C 3 BEGIN
648:C 4 IF NOT CHECKCARD(SCODE)
649:C 4 THEN BADMESSAGE('*** NO PROGRAMMER CARD IN SYSTEM ***');
650:C 3 END;
651:S
652:D -890 2 (*****
653:D 2 PROCEDURE CHECKEPROM;
654:D 3 VAR
655:D -4 3 I : INTEGER;
656:D -5 3 DONE : BOOLEAN;
657:S
658:C 3 BEGIN
659:C 3 CHECKSCODE;
660:C 3 ERROR:=EGETINFO(SCODE,EPINFO);

```

```

661:C 3 IF EPINFO.EPSTART=0 THEN
662:C 4 BADMESSAGE('NO EPROM CARD ATTACHED TO PROGRAMMER CARD');
663:S
664:C 3 EPROMDATA := EPROMLIST;
665:C 3 DONE := FALSE;
666:C 3 REPEAT
667:C 4 IF EPROMDATA=NIL THEN
668:C 5 BEGIN
669:C 5 NEW(EPROMDATA);
670:C 5 WITH EPROMDATA^, EPINFO DO
671:C 6 BEGIN
672:C 6 BASEADDR := EPSTART;
673:C 6 EPSIZE := EPEND-EPSTART;
674:C 6 EPINC := EPSIZE DIV 8;
675:C 6 ADDRESS[0] := BASEADDR;
676:C 6 PRESENT[0] := TRUE;
677:C 6 FOR I:=1 TO 7 DO
678:C 7 BEGIN
679:C 7 ADDRESS[I] := ADDRESS[I-1]+EPINC;
680:C 7 PRESENT[I] := TRUE;
681:C 7 END;
682:C 6 NEXT I := EPROMLIST;
683:C 6 EPROMLIST := EPROMDATA;
684:C 6 DONE := TRUE;
685:C 6 END;
686:C 5 END;
687:C 5 ELSE
688:C 5 BEGIN
689:C 5 IF EPROMDATA^.BASEADDR=EPINFO.EPSTART THEN DONE := TRUE
690:C 5 ELSE EPROMDATA := EPROMDATA^.NEXT;
691:C 5 END;
692:C 4 UNTIL DONE;
693:C 3 END; { CHECKEPROM }
694:S
695:D -890 2 (*****
696:D 2 PROCEDURE FINDCARD(SCODES:BOOLEAN);
697:D 3 VAR
698:D -4 3 SC : INTEGER;
699:C 3 BEGIN
700:C 3 FOR SC:=MINSC TO MAXSC DO
701:C 4 IF CHECKCARD(SC) THEN
702:C 5 BEGIN
703:C 5 SCODES THEN WRITE(SC:3);
704:C 5 SCODE := SC;
705:C 5 END;
706:C 3 END; { FINDCARD }
707:S
708:D -890 2 (*****
709:D 2 PROCEDURE DOCONFIGURE(DOPROMPT:BOOLEAN);
710:D 3 VAR
711:D -1 3 DONE : BOOLEAN;
712:D -6 3 OLDSCODE : INTEGER;
713:D -7 3 OP : CHAR;
714:D -12 3 I : INTEGER;
715:S
716:D -12 3 {-----}
717:D 3 PROCEDURE PRINTSC;
718:C 4 BEGIN
719:C 4 FGOXY(OUTPUT,0,3);
720:C 4 Writeln('Active programmer card at select code ',SCODE:1,CTEOL);

```



```

721:C      4      END;
722:S
723:S
724:D      -12   3      {-----}
725:D      3      PROCEDURE PRINTEPINF;
726:C      4      BEGIN
727:C      4      FGOTOXY(OUTPUT,0,5);
728:C      4      TRY
729:C      5      CHECKEPRDM;
730:C      5      TEMP := EPINFO.EPEND-EPINF0.EPSTART;
731:C      5      WRITELN('EPROM at address ',EPINFO.EPSTART:1,' for ',
732:C      5      TEMP:1,' bytes',CTEOL);
733:C      5      WRITELN('EPROM type XX',TEMP DIV 2048:1,CTEDL);
734:C      5      RECDVER
735:C      5      IF ESCAPECODE<>0 THEN ESCAPE(ESCAPECODE);
736:C      4      END;
737:S
738:D      -12   3      {-----}
739:D      3      PROCEDURE PRINTSOCKETS;
740:D      4      VAR
741:D      4      I : INTEGER;
742:C      4      BEGIN
743:C      4      FGOTOXY(OUTPUT,0,7);
744:C      4      WRITELN('Socket status (UL means EPROM pair present)');
745:C      4      WITH EPROMDATA^ DO
746:C      5      FOR I:= 0 TO 3 DO
747:C      6      BEGIN
748:C      6      WRITE(I:1);
749:C      6      IF PRESENT[I] THEN WRITE('UL ') ELSE WRITE(' empty');
750:C      6      WRITE(' ');
751:C      6      IF PRESENT[I+4] THEN WRITE('UL ') ELSE WRITE(' empty');
752:C      6      WRITELN;
753:C      6      END;
754:C      4      END;
755:S
756:D      -12   3      {-----}
757:D      3      BEGIN ( DOCONFIGURE )
758:C      3      FGOTOXY(OUTPUT,0,2); WRITE(CTEDS);
759:C      3      CHECKCODE;
760:C      3      OLDSOCD := SCODE;
761:C      3      WRITE('Programmer card(s) at ');
762:C      3      FINDCARD(TRUE);
763:C      3      SCODE := OLDSOCD;
764:C      3      PRINTSC;
765:C      3      PRINTBR;
766:C      3      PRINTERINFO;
767:C      3      IF EPINFO.EPSTART>0 THEN PRINTSOCKETS;
768:C      3      IF DOPROMPT THEN
769:C      4      REPEAT
770:C      5      KEY
771:C      6      PROMPTFORCHAR('CONFIGURE: Selectcode Burnrate Emptysockets Qt',OP);
772:C      6      FGOTOXY(OUTPUT,0,13); WRITE(CTEDS);
773:C      6      IF DP=SH_EXC THEN DP:='Q';
774:S
775:C      6      IF DP='S' THEN
776:C      7      BEGIN ( NEW SELECT CODE )
777:C      7      OLDSOCD := SCODE;
778:C      7      WRITE('New select code (' ,SCODE:1,' ) ? '); READNUMBER(SCODE);
779:C      7      IF (SCODE<MINSC) OR (SCODE>MAXSC) THEN
780:C      8      BEGIN SCODE := OLDSOCD; BADMESSAGE('SELECT CODE OUT OF RANGE'); END;

```

```

781:C      7      IF NOT CHECKCARD(SCODE) THEN
782:C      8      BEGIN
783:C      8      SCODE := OLDSOCD; BADMESSAGE('SELECT CODE NOT A PROGRAMMER CARD');
784:C      8      END;
785:C      8      PRINTSC;
786:C      7      PRINTBR;
787:C      7      PRINTERINFO;
788:C      7      IF EPINFO.EPSTART>0 THEN PRINTSOCKETS ELSE WRITE(CTEDS);
789:C      7      END
790:C      7      ELSE
791:C      7      IF DP='B' THEN
792:C      8      BEGIN
793:C      8      FASTBURN := NOT FASTBURN; PRINTBR;
794:C      8      END
795:C      8      ELSE
796:C      8      IF DP='E' THEN
797:C      9      BEGIN
798:C      9      IF EPINFO.EPSTART>0 THEN
799:C      10     BEGIN
800:C      10     I := -1;
801:C      10     WRITE('SOCKET (PAIR) NUMBER ? ',CTEDL); READNUMBER(I);
802:C      10     IF I>=0 THEN
803:C      11     IF (I<0) OR (I>7) THEN BADMESSAGE('SOCKET NUMBER OUT OF RANGE')
804:C      12     ELSE
805:C      12     BEGIN
806:C      12     EPROMDATA^.PRESENT[I] := NOT EPROMDATA^.PRESENT[I];
807:C      12     PRINTSOCKETS;
808:C      12     END;
809:C      10     END
810:C      10     ELSE BEEP;
811:C      9     END
812:C      9     ELSE
813:C      9     IF (DP<>'Q') THEN
814:C      10     IF STREAMING THEN BADCOMMAND(OP)
815:C      11     ELSE BEEP;
816:S
817:C      6      RECDVER
818:C      6      BEGIN
819:C      6      LOCKUP;
820:C      6      SAVEIO := IORESULT;
821:C      6      SAVEESC := ESCAPECODE;
822:C      6      IORESULT := SAVEIO;
823:C      6      IF (SAVEESC<>0) AND (SAVEESC<>-10) THEN IORESULT := ORD(INDERROR);
824:C      6      LOCKDOWN;
825:C      6      PRINTIDERRMSG;
826:C      6      FIXLOCK;
827:C      6      IF SAVEESC<>0 THEN ESCAPE(SAVEESC) ELSE DP := ' ';
828:C      6      END;
829:S
830:C      5      UNTIL DP='Q';
831:C      3      END; ( DOCONFIGURE )
832:S
833:D      -890   2      {*****}
834:D      2      PROCEDURE DOTRANSFER;
835:D      3      TYPE
836:D      3      HEADREC = PACKED ARRAY[0..17] OF BYTE;
837:S
838:D      3      HEADP = RECORD
839:D      3      CASE BOOLEAN OF
840:D      3      TRUE : (BINT:INTEGER);

```



```

841:D 3 FALSE: (BPTR: ^HEADREC);
842:D 3 END;
843:D 3
844:D -122 3 VAR FILENAME1 : FID;
845:D -244 3 SOURCEFILE : FID;
846:S
847:D -245 3 FILEMOVED : BOOLEAN;
848:D -246 3 DONE : BOOLEAN;
849:D -247 3 FORMAT : BOOLEAN;
850:S
851:D -252 3 I : INTEGER;
852:D -253 3 J : INTEGER;
853:D -260 3 EPART : INTEGER;
854:D -264 3 INSTATE : INTEGER;
855:D -268 3 OUTSTATE : INTEGER;
856:D -272 3 BUF : BIGPTR;
857:D -276 3 POSITION : INTEGER;
858:D -280 3 MOVESIZE : INTEGER;
859:D -284 3 MSIZE : INTEGER;
860:D -288 3 BUFSIZE : INTEGER;
861:D -292 3 OUTSIZE : INTEGER;
862:D -296 3 SAVEIO : INTEGER;
863:D -300 3 SAVEESC : INTEGER;
864:D -304 3 DUMWINDOW : WINDOW;
865:D -322 3 EDHEADER : HEADREC;
866:D -578 3 MSGLINE : STRING[255];
867:D -658 3 DIRCATENTRY : CATENTRY;
868:D -662 3 BLANKCHK : HEADP;
869:D -663 3 ANSWER : CHAR;
870:S
871:C 3 BEGIN ( DOIRANSFER )
872:C 3 DOCONF(URE(FALSE));
873:C 3 FGOTOXY(OUTPUT 0,13);
874:C 3 WRITELN('TRANSFER OPERATION',CTEOS);
875:C 3 CHECKEPROM;
876:S
877:C 3 WRITE('Source ('.DKVID,') ? ');
878:C 3 READLN(FILENAME1); GOODIO;
879:C 3 FILENAME1 := STRLTRIM(STRRTRIM(FILENAME1));
880:C 3 IF STRLEN(FILENAME1)=0 THEN FILENAME1:=DKVID+'.';
881:C 3 ZAPSPACES(FILENAME1);
882:C 3 IF STRLEN(FILENAME1)>0 THEN
883:C 4 WITH EPROMDATA^DO
884:C 5 BEGIN ( HAVE A SOURCE NAME )
885:C 6 WITH ININFO DO
886:C 7 BEGIN DIROPEN := FALSE; FILEOPEN := FALSE; MOUNTED := FALSE; END;
887:C 8 MARK(LHEAP); HEAPINUSE := TRUE;
888:C 9 NEWWORDS(DUMWINDOW,1); ( DUMMY WINDOW FOR FILE TRANSFER )
889:C 5 TRY
890:C 6 WITH ININFO, CFIB DO
891:C 7 BEGIN
892:C 8 ( OPEN THE SOURCE )
893:C 9 SETUPFIBFORFILE(FILENAME1,CFIB,CPVOL);
894:C 10 IF STRLEN(FTITLE)=0 THEN
895:C 11 BEGIN ( VOLUME -> EPROM )
896:C 12 USEUNIT := UNITNUMBER(CPVOL); DSTATUS := DWANTED;
897:C 13 IF USEUNIT THEN CVOL := '' ELSE CVOL := CPVOL;
898:C 14 MOUNTED := (FUNIT>0) AND NOT(UNITNUMBER(FVID));
899:C 15 IF MOUNTED THEN CVOL := FVID ELSE INMOUNT(TRUE);
900:S

```

```

901:C 8 LOCKUP; ( LOCK THE KEYBOARD THEN OPEN THE VOLUME )
902:C 8 FBUFFERED := FALSE;
903:C 8 FKIND := UNTYPEFILE; FEFT := EFTTABLE[FKIND];
904:C 8 CALL(UNITABLE[FUNIT].DAM,CFIB,FUNIT,OPENVOLUME);
905:C 8 FILEOPEN := (IDRESULT=ORD(INOERROR));
906:C 8 LOCKDOWN; ( UNLOCK THE KEYBOARD )
907:C 8 GOODIO;
908:C 8 ( FINISH THE SETUP )
909:C 8 OUTSIZE := FPEOF;
910:C 8 SOURCEFILE := '';
911:C 8 FVID := '';
912:C 8 FORMAT := FALSE;
913:C 8 FGOTOXY(OUTPUT 0,14);
914:C 8 WRITELN('TRANSFERRING VOLUME ',FVID,');
915:C 8 END ( VOLUME -> EPROM )
916:C 8 ELSE
917:C 8 BEGIN ( FILE -> EPROM )
918:C 9 OPENDIR(FILENAME1,SOURCEFILE,'SOURCE',ININFO,DIRCATENTRY);
919:C 9 IF NOT DIROPEN THEN ESCAPE(0);
920:C 9 IF STRLEN(SOURCEFILE)=0 THEN
921:C 10 BADMESSAGE('CAN'T TRANSFER A DIRECTORY');
922:C 9 FTITLE := SOURCEFILE;
923:C 9 FINITB(CFIB,DUMWINDOW,-3);
924:C 9 PATHID := PATH;
925:C 9 LOCKUP;
926:C 9 CALL(UNITABLE[FUNIT].DAM,CFIB,FUNIT,OPENFILE);
927:C 9 FILEOPEN := (IDRESULT=ORD(INOERROR));
928:C 9 LOCKDOWN;
929:C 9 GOODIO;
930:C 9 FORMAT := (FKIND=ASCIIFILE) OR (FKIND=TEXTFILE);
931:C 9 OUTSIZE := FLEOF;
932:C 9 FGOTOXY(OUTPUT 0,14);
933:C 9 WRITELN('TRANSFERRING FILE ',CVOL,','.SOURCEFILE,CTEOL);
934:C 9 END; ( FILE -> EPROM )
935:C 7 END; ( WITH ININFO, CFIB -- OPEN THE SOURCE )
936:S
937:C 6 ( ALLOCATE BUFFER SPACE )
938:C 6 BUFSIZE := (MEMAVAIL DIV 256) * 256 - 3D * 512; (SAME SDME FOR SLOP)
939:C 6 IF BUFSIZE<512 THEN ESCAPE(-2); ( NOT ENOUGH ROOM )
940:C 6 NEWWORDS(BUF,BUFSIZE DIV 2); ( ALLOCATE BUFFER SPACE )
941:S
942:C 6 ( GET START ADDRESS ON EPROM )
943:C 6 MSGLINE := '';
944:C 6 WITH EPINFO DO
945:C 7 IF SOURCEFILE='' THEN
946:C 8 BEGIN ( VOLUME TRANSFER )
947:C 9 ( SET DEFAULT START BLOCK )
948:C 10 TEMP := 0;
949:C 11 OUTSTARTA := EPSTART;
950:C 11 DONE := FALSE;
951:C 11 REPEAT
952:C 12 IF PRESENT[TEMP] THEN
953:C 13 BEGIN ( SOCKET PRESENT, CHECK CONTENTS )
954:C 14 BLANKCHK.BINT := OUTSTARTA;
955:C 15 IF (BLANKCHK.BPTR[0]=255) AND
956:C 16 (BLANKCHK.BPTR[1]=255) THEN DONE := TRUE
957:C 11 ELSE
958:C 12 BEGIN ( INCREMENT TO NEXT BLOCK )
959:C 13 OUTSTARTA := OUTSTARTA+16;
960:C 11 IF TEMP<7 THEN

```



```

961:C 11 IF OUTSTARTA>=ADDRESS[TEMP+1] THEN TEMP := TEMP+1;
962:C 11 END;
963:C 10 END;
964:C 10 ELSE
965:C 10 BEGIN
966:C 10 { SKIP EMPTY SOCKET PAIR }
967:C 10 IF TEMP<7 THEN TEMP := TEMP+1;
968:C 10 OUTSTARTA := OUTSTARTA+EPINC;
969:C 10 UNTIL 'DONE OR (OUTSTARTA>EPEND);
970:S 8
971:C 8 IF OUTSTARTA>EPEND THEN
972:C 8 BEGIN
973:C 8 Writeln('*** NO BLANK BLOCK ON THIS EPROM CARO ***');
974:C 8 OUTSTARTA:=EPSTART;
975:C 8 END;
976:C 8 OUTSTARTA := (OUTSTARTA-EPSTART) DIV K16;
977:S 8
978:C 8 WRITE('Start at EPROM block offset ('OUTSTARTA:1,') ? ');
979:C 8 READNUMBER(OUTSTARTA); OUTSTARTA:=OUTSTARTA*K16;
980:C 8 STRWRITE(MSGLINE,1,1,'BLOCK OFFSET NOT IN RANGE 0..',
981:C 8 (EPSIZE DIV K16)-1:1);
982:C 8 END;
983:C 8 ELSE
984:C 8 BEGIN { FILE TRANSFER }
985:C 8 OUTSTARTA := 0; { DEFAULT VALUE }
986:C 8 WRITE('Start at EPROM byte offset ('OUTSTARTA:1,') ? ');
987:C 8 READNUMBER(OUTSTARTA);
988:C 8 STRWRITE(MSGLINE,1,1,'BYTE OFFSET NOT IN RANGE 0..',EPSIZE-1:1);
989:C 8 END;
990:S 6
991:C 6 IF OUTSTARTA>(EPSIZE-1) THEN BADMESSAGE(MSGLINE);
992:C 6 OUTSTARTA := OUTSTARTA + EPINFO.EPSTART;
993:S 6
994:C 6 { CHECK TO SEE IF DATA WILL FIT IN AVAILABLE EPROM SPACE }
995:C 6 J := OUTSIZE / (VOL / FILE SIZE)
996:C 6 IF SOURCEFILE="" THEN
997:C 6 BEGIN
998:C 6 J := J + 18; { ADD HEADER }
999:C 6 J := J + (J DIV K16)*2; { ADD 16K HEADER GAPS }
1000:C 6 END;
1001:C 6 IF NOT CHECKSPACE(OUTSTARTA,J,1) THEN
1002:C 6 BEGIN
1003:C 6 IF I<=0 THEN
1004:C 6 BEGIN Writeln('NO EPROM AT START ADDRESS'); BADIO(INDERROR); END;
1005:S 7
1006:C 7 Writeln('DATA EXCEEDS EPROM SPACE BY ',J-1:1,' BYTES',CTEOL);
1007:C 7 IF STREAMING THEN ESCAPE(-1);
1008:C 7 PROMPTREAD('Abort transfer or Truncate file (A/T) ? ',
1009:C 7 ANSWER,'AT',SH EXC);
1010:C 7 IF ANSWER='A' THEN ESCAPE(0);
1011:C 7 IF SOURCEFILE="" THEN
1012:C 7 BEGIN
1013:C 7 OUTSIZE := (I-18); OUTSIZE := OUTSIZE - (OUTSIZE DIV K16)*2;
1014:C 7 END;
1015:C 7 ELSE OUTSIZE := I;
1016:C 7 END;
1017:S 6
1018:C 6 INSTATE := 1;
1019:C 6 PASS := 1;
1020:S

```

```

1021:C 6 BEGIN { TRY THE TRANSFER }
1022:C 6 FILEMOVED := FALSE;
1023:C 6 WITH ININFO, CFIB, EPROMDATA^ DO
1024:C 7 REPEAT { MOVE THE FILE }
1025:C 8 DONE := FALSE;
1026:C 8 REPEAT
1027:C 9 CASE INSTATE OF
1028:C 10 1: BEGIN { INITIALIZE SOURCE PARAMETERS }
1029:C 10 FGOOTOXY(OUTPUT,0,16);
1030:C 10 Writeln('now on pass ',PASS:1,CTEOL);
1031:C 10 OUTPOSITION := OUTSTARTA;
1032:C 10 LEFTTOXFER := OUTSIZE;
1033:C 10 POSITION := 0;
1034:C 10 EOF := FALSE; FEOLN := FALSE;
1035:C 10 FLASTPOS := -1; FPOS := 0;
1036:C 10 INSTATE := 2;
1037:C 10 OUTSTATE := 1;
1038:C 10 END;
1039:C 10 2: BEGIN { READ THE FILE/VOLUME }
1040:C 10 WRITE('reading ...',CTEOL,CHR(13));
1041:C 10 IF FORMAT THEN
1042:C 11 BEGIN { FORMATEO TRANSFER }
1043:C 11 ANYTOMEH(AADDR(CFIB),BUF,BUFSIZE);
1044:C 11 'DONE' := TRUE;
1045:C 11 IF FEOL THEN LEFTTOXFER := 0;
1046:C 11 GOODIO;
1047:C 11 END;
1048:C 11 ELSE
1049:C 11 BEGIN { UNFORMATEO TRANSFER }
1050:C 11 IF BUFSIZE>LEFTTOXFER THEN MOVESIZE := LEFTTOXFER
1051:C 12 ELSE MOVESIZE := BUFSIZE;
1052:C 12 CALL(UNITABLE^[FUNIT].TM,AADDR(CFIB).READBYTES,
1053:C 12 BUF^,MOVESIZE,POSITION);
1054:C 12 GOODIO;
1055:C 12 LEFTTOXFER := LEFTTOXFER - MOVESIZE;
1056:C 12 'DONE' := TRUE;
1057:C 12 END;
1058:C 10 END;
1059:C 10 CASE INSTATE )
1060:C 9 UNTIL 'DONE';
1061:C 8 WRITE(CTEOL);
1062:S 8
1063:C 8 DONE := FALSE;
1064:C 8 IF NOT FILEMOVED THEN
1065:C 8 REPEAT
1066:C 10 IF PASS=1 THEN WRITE('checking ...',CTEOL,CHR(13))
1067:C 11 ELSE WRITE('writing ...',CTEOL,CHR(13));
1068:C 10 CASE OUTSTATE OF
1069:C 11 1: BEGIN
1070:C 12 { SET BURN RATE }
1071:C 12 ERROR:=EBRATE(SCODE,FASTBURN);
1072:C 12 IF SOURCEFILE="" THEN
1073:C 12 BEGIN { VOLUME TRANSFER }
1074:C 12 { PUT EDISC VOLUME HEADER }
1075:C 12 FOR I := 0 TO 17 DO EOHEDER[I]:=0;
1076:C 12 EOHEDER[0]:=HEX('F0');
1077:C 12 EOHEDER[1]:=HEX('FF');
1078:C 12 EOHEDER[2]:=ORD(' ');
1079:C 12 EOHEDER[3]:=HEX('18');
1080:C 12 EOHEDER[12]:=HEX('01');

```



```

1081:C      12      EDHEADER[13]:=HEX('02');
1082:C      12      BURNIT(EDHEADER,18);
1083:C      12      END;
1084:C      11      OUTSTATE := 2;
1085:C      11      END;
1086:S
1087:C      11      2: BEGIN { WRITE DATA }
1088:C      11      IF FORMAT THEN MEMTOEPROM(BUF)
1089:C      11      ELSE
1090:C      12      BEGIN
1091:C      12      IF SOURCEFILE='' THEN
1092:C      13      BEGIN { TRANSFERING A VOLUME }
1093:C      13      I := 0;
1094:C      13      REPEAT
1095:C      14      { WATCH FOR 16K BYTE BOUNDARIES }
1096:C      14      MSIZE:=((OUTPOSITION*K16) DIV K16)-OUTPOSITION;
1097:C      14      IF MSIZE>MOVESIZE THEN MSIZE:=MOVESIZE;
1098:C      14      BURNIT(BUF[I],MSIZE);
1099:C      14      I := I + MSIZE;
1100:C      14      MOVESIZE := MOVESIZE - MSIZE;
1101:C      14      IF ((OUTPOSITION MOD K16)=0) AND
1102:C      15      ((MOVESIZE>0) OR (LEFTTOXFER>0)) THEN
1103:C      15      BEGIN { PUT ZEROES IN BOUNDARY BYTES }
1104:C      15      J:=0; BURNIT(J,2);
1105:C      15      END;
1106:C      14      UNTIL MOVESIZE=0;
1107:C      13      END
1108:C      13      ELSE
1109:C      13      BEGIN { TRANSFERING A FILE }
1110:C      13      BURNIT(BUF^,MOVESIZE);
1111:C      13      END;
1112:C      12      DONE:=TRUE;
1113:C      11      IF LEFTTOXFER=0 THEN
1114:C      11      BEGIN
1115:C      12      IF PASS=2 THEN FILEMOVED:=TRUE
1116:C      13      ELSE
1117:C      13      BEGIN PASS :=2; INSTATE := 1; END;
1118:C      13      END;
1119:C      12      END;
1120:C      11      { CASE OUTSTATE }
1121:C      11      UNTIL DONE;
1122:C      10      WRITE(CTEOL);
1123:C      8      UNTIL FILEMOVED;
1124:C      8      Writeln('TRANSFER COMPLETED');
1125:C      6      IF FORMAT THEN I := OUTPOSITION-OUTSTARTA
1126:C      6      ELSE I := OUTSIZE;
1127:C      7      Writeln(I:1,' data bytes programmed and verified');
1128:C      6      END;
1129:C      6      RELEASE(LHEAP);      HEAPINUSE := FALSE;
1130:C      6      CLOSEALL;
1131:C      6      RECOVER
1132:C      6      BEGIN
1133:C      6      LOCKUP;
1134:C      6      RELEASE(LHEAP);      HEAPINUSE := FALSE;
1135:C      6      SAVEIO := IORESULT;
1136:C      6      SAVEESC := ESCAPECODE;
1137:C      6      CLOSEALL;
1138:C      6      IORESULT := SAVEIO;
1139:C      6      LOCKDOWN;
1140:C      6

```

```

1141:C      6      PRINTIDERRMSG;
1142:C      6      IF SAVEESC<>D THEN ESCAPE(SAVEESC);
1143:C      6      END;
1144:C      6      END; { HAVE SOURCE NAME }
1145:C      3      END; { DTRANSFER }
1146:D      -890  2
1147:D      -890  2 (*****
1148:D      -82  2 PROCEDURE PUTMENU(MSTRING:STRING80);
1149:C      3      BEGIN
1150:C      3      FGOTOXY(OUTPUT,D,2);
1151:C      3      WRITE(MSTRING,'?',CTEOL);
1152:C      3      END;
1153:D      -890  2 (*****
1154:D      -890  2 PROCEDURE DOBLANKCHECK;
1155:D      3      TYPE
1156:D      3      TWOBYTES = PACKED ARRAY[0..1] OF BYTE;
1157:D      3      VAR
1158:D      3      OLDSTART : INTEGER;
1159:D      -4  3      START : INTEGER;
1160:D      -9  3      ENDSCAN : INTEGER;
1161:D      -12 3      NBYTES : INTEGER;
1162:D      -18 3      BLANKS : BOOLEAN;
1163:D      -17 3      LINES : INTEGER;
1164:D      -22 3      I : INTEGER;
1165:D      -28 3      X,Y : INTEGER;
1166:D      -34 3      BREC : RECORD
1167:D      -34 3      CASE BOOLEAN OF
1168:D      -34 3      TRUE:(BPTR : ^TWOBYTES);
1169:D      -34 3      FALSE:(BINT : INTEGER);
1170:D      -34 3      END;
1171:D      -38 3      BEGIN
1172:C      3      DOCONFIGURE(FALSE);
1173:C      3      FGOTOXY(OUTPUT,0,13);
1174:C      3      Writeln('BLANK CHECK',CTEOL);
1175:C      3      CHECKEPROM;
1176:C      3      OLDSTART := 0;
1177:C      3      BLANKS := FALSE;
1178:C      3      FGOTOXY(OUTPUT,X,Y); LINES := 0;
1179:C      3      WITH EPROMDATA DO
1180:C      3      FOR I:=0 TO 7 DO
1181:C      4      IF NOT PRESENT[I] THEN
1182:C      5      BEGIN { CLOSE OFF REPORT OF PREVIOUS PAIR }
1183:C      6      IF BLANKS THEN
1184:C      6      Writeln(BREC.BINT-1-BASEADDR:1,' (' ,BREC.BINT-OLDSTART:1,')');
1185:C      7      BLANKS:=FALSE
1186:C      6      END
1187:C      6      ELSE
1188:C      6      BEGIN { SOCKET PAIR PRESENT SO CHECK IT OUT }
1189:C      6      BREC.BINT := ADDRESS[I];
1190:C      6      ENDSCAN := BREC.BINT + EPINC;
1191:C      6      REPEAT
1192:C      7      IF BLANKS THEN
1193:C      8      BEGIN
1194:C      8      IF BREC.BPTR^[0]<>255 THEN
1195:C      9      BEGIN
1196:C      9      Writeln(BREC.BINT-1-BASEADDR:1,' (' ,BREC.BINT-OLDSTART:1,')');
1197:C      9      BLANKS:=FALSE;
1198:C      9      END;
1199:C      8      END;
1200:C      8      END

```



```

1201:C      ELSE
1202:C      BEGIN
1203:C      IF BREC.BPTR^[0]=255 THEN
1204:C      BEGIN
1205:C      IF LINES>5 THEN
1206:C      BEGIN
1207:C      LINES := 0; SPACEWAIT;
1208:C      FGOTOXY(OUTPUT,X,Y); WRITE(CTEOS);
1209:C      END;
1210:C      LINES := LINES + 1;
1211:C      OLDSTART := BREC.BINT; WRITE(OLDSTART-BASEADDR,' - ');
1212:C      ELANKS:=TRUE;
1213:C      END;
1214:C      END;
1215:C      BREC.BINT := BREC.BINT + 1;
1216:C      UNTIL EREC.BINT=ENDSCAN;
1217:C
1218:C      IF BLANKS AND (I=7) THEN
1219:C      WRITELN(BREC.BINT-1-BASEADDR:1,' ',BREC.BINT-OLDSTART:1,'');
1220:C
1221:C      END; { FOR I := ... }
1222:C      IF OLDSTART=0 THEN WRITELN('NO BLANK SPACE FOUND');
1223:C      END; { DOBLANKCHECK }
1224:C
1225:C      -B90 2 (*****
1226:C      BEGIN { COMMANDLEVEL }
1227:C      FIXLOCK;
1228:C      WITH ININFO DO BEGIN DIOPEN:=FALSE; FILEOPEN:=FALSE; END;
1229:C      HEAPINUSE := FALSE;
1230:C      IORESULT := ORD(INOERROR);
1231:C      SCODE := 0; TEMP := 0;
1232:C      FASTBURN := FALSE; { DEFAULT BURN RATE }
1233:C      EPROMLIST := NIL; { NO EPROM CARD INFO YET }
1234:C      FINDCARD(FALSE); { FIND A PROGRAMMER CARD }
1235:C      TRY
1236:C      DOCONFIGURE(FALSE); { DISPLAY DEFAULT CONFIGURATION }
1237:C      RECOVER
1238:C      IF ESCAPECODE<>0 THEN ESCAPE(ESCAPECODE);
1239:C
1240:C      FGOTOXY(OUTPUT,0,13);
1241:C      WRITELN('Copyright 1983 Hewlett-Packard Company. ');
1242:C      WRITELN(' All rights are reserved. ');
1243:C
1244:C      REPEAT
1245:C      TRY
1246:C      PROMPTFORCHAR('ETU: Transfer Configure Blankcheck Quit',OP);
1247:C      IF OP=SH_EXC THEN OP:='Q';
1248:C
1249:C      IF OP='T' THEN DOTRANSFER
1250:C      ELSE
1251:C      IF OP='C' THEN DOCONFIGURE(TRUE)
1252:C      ELSE
1253:C      IF OP='B' THEN DOBLANKCHECK
1254:C      ELSE
1255:C      IF OP='Q' THEN BEGIN END
1256:C      ELSE
1257:C      IF STREAMING THEN BADCOMMAND(OP)
1258:C      ELSE BEEP;
1259:C
1260:C      RECOVER
1261:C      BEGIN

```

```

1261:C      4 LOCKUP;
1262:C      4 IF HEAPINUSE THEN RELEASE(LHEAP);
1263:C      4 HEAPINUSE := FALSE;
1264:C      4 SAVEIO := IORESULT;
1265:C      4 SAVEESC := ESCAPECODE;
1266:C      4 CLOSEALL;
1267:C      4 IORESULT := SAVEIO;
1268:C      4 IF (SAVEESC<>0) AND (SAVEESC<>-10) THEN IORESULT := ORD(INOERROR);
1269:C      4 LOCKDOWN;
1270:C      4 PRINTIOERRMSG;
1271:C      4 FIXLOCK;
1272:C      4 IF SAVEESC<>0 THEN ESCAPE(SAVEESC) ELSE OP := ' ';
1273:C      4 END;
1274:C      3 UNTIL OP='Q';
1275:C      2 END {COMMANDLEVEL} ;
1276:C
1277:C      1 (*****
1278:C      1 BEGIN
1279:C      1 WRITELN(CLEARSCR);
1280:C      1 FGOTOXY(OUTPUT,0,1);
1281:C      1 WRITELN('EPROM TRANSFER UTILITY (4-Jun-84)');
1282:C      1 COMMANDLEVEL;
1283:C      1 END.
1284:C

```

No errors. No warnings.

***** Nonstandard language features enabled *****

F9885

Description

F9885 is the transfer method (TM) for the 9885 flexible disc drives.

Requirements

IODECLARATIONS, GPIODVR.

Notes

Although the 9885 drive uses the 98622 GPIO interface and the 98620 DMA card, it does not require the IOLIBRARY GPIO and DMA modules.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1983.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S     RESTRICTED RIGHTS LEGEND
11:S
12:S     Use, duplication, or disclosure by the Government
13:S     is subject to restrictions as set forth in
14:S     paragraph (b) (3) (B) of the Rights in Technical
15:S     Data and Computer Software clause in
16:S     DAR 7-104.9(a).
17:S
18:S     HEWLETT-PACKARD COMPANY
19:D     0 Fort Collins, Colorado      *)
20:S
21:S
22:D     0 $modals
23:D     0 $debug off, range off, ovfcheck off$
24:D     0 $stackcheck off, ioccheck off$
25:S
26:D     0 $search 'GPIODVR', 'IOLIB:COMASM', 'IOLIB:KERNEL'$
27:S

```

```

28:D     0 $pages
29:D     0 $copyright 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'S
30:S
31:D     0 program F9885init;
32:S
33:S
34:D     1 module F9885dvr;
35:S
36:D     1 import
37:D     1   sysglobals, mini, gp, iodeclarations, iocomasm, misc;
38:S
39:D     1 export
40:D     1   procedure F9885io (fp: fibp; request: amrequesttype; anyvar buffer: window;
41:D     2   length, position: integer);
42:D     1 implement
43:S
44:S
45:D     1 procedure F9885io;
46:S
47:D     2   type
48:D     2   errors = (noerror, nopower, dooropen, nodisc, badcommand, norecord,
49:D     2   notrack, badcheckword, dataoverrun, badverify);
50:S
51:D     2   primarycommands = (readblock, verifyblock, writeblock, settracksector);
52:S
53:D     2   fd = (floppy disc command & status structure)
54:D     2   packed record case integer of
55:D     2   -1: (w: shortint);
56:D     2   0: (case primary: primarycommands of
57:D     2   readblock, verifyblock, writeblock:
58:D     2   (drv: 0..3; nrecords: 0..4095);
59:D     2   settracksector:
60:D     2   (drv: 0..3; track: 0..127; sector: 0..31));
61:D     2   1: (pad: 0..15; errcode: errors; p2, transfercomplete,
62:D     2   seekcomplete, notready, writeprotected, dooropened: boolean;
63:D     2   drve: 0..3);
64:D     2   end;
65:S
66:D     2   gpio_enable_type = packed array[0..1] of gpio_r3_type;
67:S
68:D     2   const
69:D     2   maxtries = 10;
70:D     2   password = -20857;
71:S
72:D     2   gpio_enable = (gpio enable bytes for the 2 DMA channels)
73:D     2   gpio_enable_type
74:D     2   [ gpio_r3_type
75:D     2   [ Wena5: false, W3pad:0, Wword:true, Wdmac1:false, Wdmac0:true ],
76:D     2   gpio_r3_type
77:D     2   [ Wena5: false, W3pad:0, Wword:true, Wdmac1:true, Wdmac0:false ] ];
78:D     2   var
79:D     -4 2   uep: ^unitentry;
80:D     -8 2   gpitr: ^gpitype;
81:D     -12 2   iptr: pio_tmp_ptr;
82:D     -16 2   bufptr: char_ptr;

```



```

83:D -16 2 $page$
84:S
85:D 2 procedure clear_unit;
86:C begin
87:C with gp^ do
88:C if sti0 or sti0 then
89:C ioresc(znodevice);
90:C gp^clear(gp^); (also tests psts while waiting for ready)
91:C 3 end;
92:S
93:S
94:D 2 procedure clear_and_escape(escape_value: shortint; iores_value: integer);
95:C begin {clear_and_escape}
96:C try
97:C gp^clear(gp^);
98:C recover
99:C {do nothing};
100:C ioresult := iores_value;
101:C escape(escape_value);
102:C 3 end; {clear_and_escape}
103:S
104:S
105:D 2 procedure transfer(record_addr, total_words: integer);
106:D var
107:D gp^dma_proc: procedure(var gp^: gp^type;
108:D command: shortint; enable_byte: gp^_r3_type;
109:D var dma_channel: dmachanneltype;
110:D buffer: charptr; length: integer);
111:D -8
112:D -12 3 status, opcode: fd;
113:D -18 3 chan, tries, sectors: shortint;
114:D -22 3 words: integer;
115:D -22 3 const
116:D -22 3 request_status = fd
117:D -22 3 [ primary: settracksector, driv: 0, track: 127, sector: 31 ];
118:D 3 begin {transfer}
119:D 3 if not dma_here then ioresc(zbaddma);
120:S
121:C 3 gp^_.r3 := 0; (setup gp^ card)
122:C 3 gp^_.r7 := 0;
123:S
124:C 3 gp^wordout(gp^, password); (issue password)
125:C 3 opcode := request_status;
126:C 3 opcode.driv := uep^.du;
127:C 3 gp^wordout(gp^, opcode.w); (issue request status command)
128:C 3 gp^wordout(gp^, 0); (clear output regs & request data word)
129:C 3 status.w := gp^wordin(gp^); (input status word)
130:S
131:C 3 if (status.driv<>uep^.du) or (status.pad<>0) then
132:C 4 clear_and_escape(-10, ord(zcatchall));
133:S
134:C 3 if status.dooropened then
135:C 4 begin
136:C 4 uep^.umediavalid := false;
137:C 4 if uep^.ureportchange then
138:C 5 ioresc(zmediumchanged);
139:C 4 end; {if}

```

```

140:C 3 $page$
141:S
142:C 3 tries := 0;
143:C 3 while total_words>0 do
144:C 4 begin
145:C 4 try
146:C 4 gp^wordout(gp^, password);
147:C 4 opcode.primary := settracksector;
148:C 4 opcode.driv := uep^.du;
149:C 4 opcode.track := record_addr div 30;
150:C 4 opcode.sector := record_addr mod 30;
151:C 4 gp^wordout(gp^, opcode.w);
152:S
153:C 4 repeat
154:C 4 chan := dma_request(tptr);
155:C 4 until chan=0;
156:C 4 if (chan<>0) and (chan<>1) then ioresc(zcatchall);
157:S
158:C 4 if total_words<=65536
159:C 4 then words := total_words
160:C 4 else words := 65536;
161:C 4 sectors := (words+127) div 128;
162:S
163:C 4 gp^wordout(gp^, password);
164:C 4 opcode.driv := uep^.du;
165:C 4 opcode.nrecords := sectors;
166:C 4 case request of
167:C 4 readbytes, startread:
168:C 4 begin
169:C 4 opcode.primary := readblock;
170:C 4 gp^dma_proc := gp^dma_in;
171:C 4 end;
172:C 4 writebytes, startwrite:
173:C 4 begin
174:C 4 opcode.primary := writeblock;
175:C 4 gp^dma_proc := gp^dma_out;
176:C 4 end;
177:C 4 end; {case}
178:C 4 call(gp^dma_proc, gp^, opcode.w, gp^_enable[chan], dma_port[chan], bufptr, words);
179:C 4 ioresc(inoerror); {invoke proper cleanup}
180:C 4 recover
181:C 4 begin
182:C 4 gp^_.r3 := 0; (disable the gp^ card)
183:C 4 dma_release(tptr); (release the dma resource)
184:C 4 if (escapecode<10) and
185:C 4 ((ioresult=ord(inoerror)) or (ioresult=ord(zcatchall)))
186:C 4 then ioresult := ord(inoerror)
187:C 4 else clear_and_escape(escapecode, ioresult);
188:C 4 end; {recover}
189:S
190:S
191:C 4 with gp^ do
192:C 4 begin
193:C 4 r7 := 1; (set the end of transfer bit)
194:C 4 udata := 0; (clear bidirectional buffer for reading status)
195:C 4 setpctl := 0; (request the status word)
196:C 4 status.w := gp^wordin(gp^); (save the status word)
197:C 4 r7 := 0; (clear the end of transfer bit)
198:C 4 end; {with}

```



```

198:C 4 $page$
200:S
201:C 4 if (status.drive<>uep^.du) or (status.pad<>0) then
202:C 1 clear_and_escape(-10, ord(zcatchall));
203:S
204:C 4 with status do
205:C 5 case errcode of
206:C 5 noerror:
207:C 6 begin
208:C 7 if notready or (not seekcomplete) or (not transfercomplete) then
209:C 6 clear_and_escape(-10, ord(zcatchall));
210:C 6 tries := 0;
211:C 6 record_addr := record_addr+sectors;
212:C 6 total_words := total_words+words;
213:C 6 bufptr := addr(bufptr^, words*2)
214:C 6 end;
215:S
216:C 6 nopower:
217:C 6 ioresc(znodevice);
218:S
219:C 5 dooropen, nodisc:
220:C 6 ioresc(znomedium);
221:S
222:C 6 badcommand:
223:C 7 if writeprotected and ((request=writebytes) or (request=startwrite))
224:C 7 then ioresc(zprotected)
225:C 7 else clear_and_escape(-10, ord(zcatchall));
226:S
227:C 6 notrack:
228:C 6 ioresc(znoblock);
229:S
230:C 6 norecord, badcheckword:
231:C 6 begin
232:C 6 tries := tries+1;
233:C 6 if tries>maxtries then
234:C 7 begin
235:C 7 if errcode=norecord then ioresc(znoblock);
236:C 7 if errcode=badcheckword then ioresc(zbadblock);
237:C 7 ioresc(zcatchall);
238:C 7 end; (if)
239:C 6 end;
240:S
241:C 6 dataoverrun:
242:C 6 ioresc(zbadhardware);
243:S
244:C 6 otherwise
245:C 6 clear_and_escape(-10, ord(zcatchall));
246:C 6 end; (case)
247:S
248:C 4 end; (while)
249:S
250:C 3 end; (transfer)

```

```

251:D -16 2 $page$
252:S
253:C 2 begin {F9885io}
254:C 2 uep := addr(unitable^[fp^.funit]);
255:C 2 if uep^.offline then ioresult := ord(znodevice)
256:C 2 else
257:C 3 begin
258:C 3 lockup;
259:C 3 try
260:C 4 with isc_table[uep^.sc] do
261:C 5 begin
262:C 5 if card_id<>hp98622 then ioresc(znodevice);
263:C 5 goir := card_ptr;
264:C 5 totir := io_tmp_ptr;
265:C 5 end; (with)
266:S
267:C 4 case request of
268:C 5 clearunit:
269:C 5 clear_unit;
270:C 5
271:C 5 unitstatus:
272:C 5 fp^.fbusy := false;
273:C 5
274:C 5 flush:
275:C 5 (do nothing);
276:C 5
277:C 5 readbytes, writebytes, startread, startwrite:
278:C 5 begin
279:C 5 if uep^.ureportchange and not uep^.umediavalid then
280:C 5 ioresc(zmediumchanged);
281:C 5 bufptr := addr(buffer);
282:C 5 if (position mod 256<>0) or odd(integer(bufptr)) then
283:C 5 ioresc(zbadmode);
284:C 5 if (position<0) or (length<0) or (position+length>fp^.fpeof) then
285:C 5 ioresc(zeof);
286:C 5 transfer((position+fp^.fileid+uep^.byteoffset) div 256, (length+1) div 2);
287:C 5 end;
288:S
289:C 5 otherwise
290:C 5 ioresc(zbadrequest);
291:C 5 end; (cases)
292:S
293:C 4 ioresc(inoerror); (set ioresult & perform lockdown)
294:C 4 recover
295:C 4 begin
296:C 4 lockdown;
297:C 4 if escapecode<>-10 then escape(escapecode);
298:C 4 if (request=startread) or (request=startwrite) then call(fp^.feof, fp);
299:C 4 end; (recover)
300:C 4 end; (else)
301:C 2 end; {F9885io}
302:S
303:C 1 end; {F9885dvr}
304:S
305:S
306:D 1 < program F9885init >
307:D 1 < import
308:D 1 < loader;
309:D
310:S

```



```
311:C      1 begin (F9885init)
312:C      1 markuser;
313:C      1 end. (F9885init)
314:S
315:S
```

No errors. No warnings.

***** Nonstandard language features enabled *****

GCRT

Description

GCRT contains high-level code for the high-resolution 9837A CRT, including the TM, the debugger window handler, and the dump graphics procedure.

Requirements

SYSGLOBALS, ASM, MISC, and SYSDEVS.

Notes

This module is linked with GASSM (in the assembly listings) to produce the INITLIB module CRTB.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

121:D -18 1 procedure cclear(x,y,n:shortint);external;
122:D -18 1 procedure cupdatecursor(x,y:shortint);external;
123:D -18 1 procedure cbuidtable;external;
124:D -18 1 procedure cshiftleft; external;
125:D -18 1 procedure cshiftright; external;
126:D -18 1 procedure cexchange( savearea: windowp; ymin, ymax, xmin, width: shortint);
127:D -18 2 external;
128:D -18 1 procedure cscrollwindow( ymin, ymax, xmin, width: shortint); external;
129:D -18 1 procedure cscrollwindn( ymin, ymax, xmin, width: shortint); external;
130:D -18 1 procedure cdbscroll( ymin, ymax, xmin, width: shortint); external;
131:D -18 1 procedure cdbscrollr( ymin, ymax, xmin, width: shortint); external;
132:D -18 1 procedure cdbhigh( c, x, y: shortint); external;
133:D -18 1
134:D 1 procedure dumpg ;
135:S
136:S
137:S
138:S
139:D 2 const
140:D 2 gwidthb = 128;
141:D 2 gmaxheight = 512;
142:D 2 gbufferize = gwidthb + 7;
143:S
144:D 2 type
145:D 2 gbyte = 0..255;
146:D 2 row_def = packed array [0..(1024*768)-1] of gbyte;
147:S
148:D 2 var
149:D -4 2 row : ^row_def;
150:S
151:D -140 2 gbuffer : packed array [1..gbufferize] of char;
152:D -148 2 i,j : integer;
153:D -152 2 index : integer;
154:D -156 2 bit_mask : integer;
155:D -160 2 result : integer;
156:S
157:C 2 begin
158:S
159:C 2 row := anyptr(frameaddr);
160:S
161:C 2 write(gfiles[4]^,#27*'A'); ( initiate graphics sequence )
162:S
163:C 2 gbuffer[1] := chr(27); ( escape sequence for graphics )
164:C 2 gbuffer[2] := '*';
165:C 2 gbuffer[3] := 'b';
166:C 2 gbuffer[4] := '1';
167:C 2 gbuffer[5] := '8';
168:C 2 gbuffer[6] := '8';
169:C 2 gbuffer[7] := 'w';
170:S
171:C 2 for j := 0 to 767 do
172:C 3 begin
173:C 4 for i := 0 to 127 do
174:C 5 begin
175:C 6 result := 0;
176:C 6 index := j*1024+i*8;
177:C 6 bit_mask := 256;
178:C 6 for index := index to index+7 do
179:C 7 begin
180:C 8 bit_mask := bit_mask div 2;

```

```

181:C 5 if odd(row^[index]) then result := bit_mask*result;
182:C 5 end;
183:C 5 gbuffer[i+8] := chr(result);
184:C 4 end;
185:C 3 write(gfiles[4]^,gbuffer:gwidthb+7);
186:C 3 if ioreult <> ord(ioerror) then goto 1;
187:C 3 end;
188:S
189:C 2 write(gfiles[4]^,#27*'B'); ( terminate graphics sequence )
190:C 2 1;
191:C 2 end;
192:S
193:S
194:S
195:S
196:D 1 procedure doupdatecursor;
197:D -2 var cursaddr: crtcmdwr;
198:C 2 begin
199:C 2 cupdatecursor(xpos,ypos);
200:C 2 end;
201:S
202:D 1 procedure getxy(var x,y: integer);
203:C 2 begin
204:C 2 x := xpos; y := ypos;
205:C 2 end;
206:S
207:D 1 procedure setxy(x, y: shortint);
208:C 2 begin
209:C 2 if x>=screenwidth then xpos:=maxx
210:C 3 else if x<0 then xpos:=0
211:C 4 else xpos := x;
212:C 2 if y>=screenheight then ypos:=maxy
213:C 3 else if y<0 then ypos:=0
214:C 4 else ypos := y;
215:C 2 end;
216:S
217:D 1 procedure gotoxy(x,y: integer);
218:C 2 begin
219:C 2 setxy(x,y);
220:C 2 doupdatecursor;
221:C 2 end;
222:S
223:S
224:D 1 procedure clear(number: shortint); { REVISED FOR 3.01 }
225:D -4 var x,y: shortint;
226:D -6 2 clearchars: shortint;
227:C 2 begin
228:C 2 x:=xpos; y:=ypos;
229:C 2 while number>0 do begin
230:C 3 if maxx-x+1<number then
231:C 4 clearchars:=maxx-x+1
232:C 4 else
233:C 4 clearchars:=number;
234:C 3 cclear(x,y,clearchars);
235:C 3 number:=number-clearchars;
236:C 3 x:=0; if y<maxy then y:=y+1;
237:C 3 end;
238:C 2 end;
239:S
240:D 1 procedure scrollup;

```



```

241:C      2 begin
242:C      2   cscrollup;
243:C      2 end;
244:S
245:D      1 procedure scrollldown;(new 4/30/81)
246:C      2 begin
247:C      2   cscrollldown;
248:C      2 end;
249:S
250:S
251:D      1 function maptcrt(c:char):shortint;
252:S
253:S      { Converts Katakana codes to their correct CRT font storage codes.
254:D      2 Note that the Yen symbol overlays the USASCII backslash (\). }
255:D      2 procedure mapkanatocrt ;
256:D      3 const
257:D      3   yenromlocation = 188; { location of Yen symbol in font storage }
258:D      3   yencode=92;
259:D      3 begin
260:C      4   if ord(c) = yencode then maptcrt := yenromlocation
261:C      4   else if ord(c)<128 then maptcrt:=ord(c)
262:C      5   else maptcrt:= ord(c)+128;
263:C      3 end; { mapkanatocrt }
264:S
265:C      2 begin
266:C      2   if kbdlang = katakana_kbd then mapkanatocrt
267:C      3   else maptcrt:=ord(c);
268:C      2 end;
269:S
270:S
271:S
272:D      1 procedure docrtio(fp: fibp; request: amrequesttype; anyvar buffer: window;
273:D      2 length, position: integer);
274:D      -1 var c: char;
275:D      -4   s: string[1];
276:D      -8   buf: charptr;
277:C      2 begin
278:C      2   ioresult := ord(ioerror);
279:C      2   buf := addr(buffer);
280:C      2   case request of
281:C      3   (uwait:
282:C      3   setcursor: gotoxy(fp^.fxpos, fp^.fypos);
283:C      3   getcursor: getxy (fp^.fxpos, fp^.fypos);
284:C      3   flush: {do nothing};
285:C      3   urtstatus: kbdio(fp, request, buffer, length, position);
286:C      3   clearunit: highlight := defaulthighlight;
287:C      3   readtoeol:
288:C      4   begin
289:C      4   buf := addr(buf^, 1);
290:C      4   buffer[0] := chr(0);
291:C      4   while length>0 do
292:C      5   begin
293:C      5   kbdio(fp, readtoeol, s, 1, 0);
294:C      5   if strlen(s)=0 then length := 0
295:C      5   { else if s[1] = chr(etc) then length := 0 }
296:C      5   else begin
297:C      5   length := length - 1;
298:C      5   crtio(fp, writebytes, s[1], 1, 0);
299:C      5   buf := addr(buf^, 1);
300:C      5   buffer[0] := chr(ord(buffer[0])+1);

```

```

301:C      5   end;
302:C      4   end;
303:C      3   end;
304:C      3   startread,
305:C      3   readbytes:
306:C      3   begin
307:C      3   while length>0 do
308:C      4   begin
309:C      4   kbdio(fp, readbytes, buf^, 1, 0);
310:C      4   if buf^ = chr(etc) then length := 0
311:C      5   else length := length - 1;
312:C      4   if buf^ = eol then crtio(fp, writeeol, buf^, 1, 0)
313:C      5   else crtio(fp, writebytes, buf^, 1, 0);
314:C      4   buf := addr(buf^, 1);
315:C      4   end;
316:C      3   if request = startread then call(fp^.feot, fp);
317:C      3   end;
318:C      3   writeeol: begin
319:C      3   if ypos=maxy then scrollup;
320:C      3   gotoxy(0, ypos+1);
321:S
322:C      3   end;
323:C      3   startwrite,
324:C      3   writebytes:
325:C      3   begin
326:C      3   while length>0 do
327:C      4   begin
328:C      4   c:=buf^; buf:=addr(buf^,1); length:=length-1;
329:S
330:C      4   case c of
331:C      5   homechar: setxy(0,0);
332:C      5   leftchar: if (xpos = 0) and (ypos>0) then setxy(maxx, ypos-1)
333:C      6   else setxy(xpos-1, ypos);
334:C      5   rightchar: if (xpos = maxx) and (ypos<maxy) then setxy(0, ypos+1)
335:C      6   else setxy(xpos+1, ypos);
336:C      5   upchar: begin if ypos <= 1 then scrollldown;
337:C      6   if ypos>0 then setxy(xpos, ypos-1);
338:C      5   end;
339:C      5   downchar: if ypos=maxy then scrollup
340:C      6   else setxy(xpos, ypos+1);
341:C      5   bellchar: beep;
342:C      5   cteos: clear(screenwidth-xpos);
343:C      5   clearscr: begin setxy(0,0); clear(screenwidth); end;
344:C      5   eol: setxy(0, ypos);
345:C      5   chr(etc): length:=0;
346:C      5   otherwise if (ord(c)>=128) and (ord(c)<144) then
347:C      6   highlight:= (ord(c)-128)*256
348:C      6   else
349:C      6   begin
350:C      6   changecursor;
351:C      6   cchar(maptcrt(c),xpos,ypos);
352:C      6   changecursor;
353:C      6   if xpos = maxx then
354:C      7   begin
355:C      7   if ypos = maxy then scrollup;
356:C      7   setxy(0, ypos+1);
357:C      7   end
358:C      7   else setxy(xpos+1, ypos);
359:C      7   end;
360:C      6   end;

```



```

361:C      5      end;
362:C      4      doupdatecursor;
363:C      4      end; (while)
364:C      3      if request = startwrite then call(fp^.feot, fp);
365:C      3      end;
366:C      3      otherwise ioresult := ord(ibadrequest);
367:C      3      end; (case)
368:C      2      end;
369:C
370:D      1      procedure lineops(op: crtllops; anyvar position: integer; c: char);
371:S
372:D      -4      2      var i, j: shortint;
373:D      -8      2      sptr: ^string255;
374:S
375:C      2      begin
376:C
377:C      2      j:=highlight; highlight:=defaulthighlight;
378:C
379:C      2      case op of
380:C
381:C      3      cllput: cchar(maptocrt(c), position, screenheight);
382:C
383:C      3      cllshiftl:
384:C      3      begin
385:C      3      cshiftleft;
386:C      3      cchar(ord(' '), maxx-8, screenheight);
387:C      3      end;
388:C
389:C      3      cllshiftr:
390:C      3      begin
391:C      3      cshiftright;
392:C      3      cchar(ord(' '), 0, screenheight);
393:C      3      end;
394:C
395:C      3      cllclear: cclear(0, screenheight, maxx-7);
396:C
397:C      3      clldisplay:
398:C      3      begin
399:C      3      sptr:=addr(position);
400:C      3      for i:=1 to strlen(sptr^) do
401:C      3      cchar(maptocrt(sptr[i]), i-1, screenheight);
402:C      3      for i:=strlen(sptr) to (maxx-8) do
403:C      3      cchar(ord(' '), i, screenheight);
404:C      3      end;
405:C
406:C      3      putstatus: cchar(ord(c), maxx-7+position, screenheight);
407:C
408:C      3      end; (of case)
409:C      2      highlight:=j;
410:C
411:C      2      end;
412:S
413:D      1      procedure crtdebug(op: dbcrtops; var dbrec: dbcinfor);
414:S
415:D      2      type
416:D      2      iptr = ^iarray;
417:D      2      iarray = array[0..maxint] of shortint;
418:S
419:D      -2      2      var i: shortint;
420:D      -6      2      j: integer;

```

```

421:D      -10      2      tempaddr: integer;
422:S
423:C      2      begin
424:C      2      with dbrec do begin
425:C      3      case op of
426:C
427:C      4      dbinfo: savesize:=(xmax-xmin+1)*(ymax-ymin+1)*128; (assumes 8x14 char)
428:C
429:C      4      dbgotoxy: cupdatecursor(cursx, cursy);
430:C
431:C      4      dbscrollup: cscrollwindow( ymin, ymax, xmin, xmax-xmin+1);
432:C
433:C      4      dbscrollldn: cscrollwinddn(ymin, ymax, xmin, xmax-xmin+1);
434:C
435:C      4      dbscrollll: begin
436:C      4      cdbscrollll(ymin, ymax, xmin, xmax-xmin+1);
437:C      4      changecursor;
438:C      4      for i:=ymin to ymax do
439:C      5      cchar(ord(' '), xmax, i);
440:C      4      changecursor;
441:C      4      end;
442:C
443:C      4      dbscrollrr: begin
444:C      4      cdbscrollrr(ymin, ymax, xmin, xmax-xmin+1);
445:C      4      changecursor;
446:C      4      for i:=ymin to ymax do
447:C      5      cchar(ord(' '), xmin, i);
448:C      4      changecursor;
449:C      4      end;
450:C
451:C      4      dbhighl: cdbhighl( ord(c), cursx, cursy);
452:C
453:C      4      dbput: begin
454:C      4      changecursor;
455:C      4      i:=highlight; highlight:=defaulthighlight;
456:C      4      cchar(ord(c), cursx, cursy);
457:C      4      highlight:=i;
458:C      4      changecursor;
459:C      4      end;
460:C
461:C      4      dbclear:
462:C      4      for i:=ymin to ymax do
463:C      5      cclear( xmin, i, xmax-xmin+1);
464:C
465:C      4      dbcline: cclear( cursx, cursy, xmax-cursx+1);
466:C
467:C      4      dbinit:
468:C      4      begin
469:C      4      for j:= 0 to (savesize div 2)-1 do
470:C      5      iptr(savearea)[j]:=0;
471:C      4      cursx:=xmin; cursy:=ymin;
472:C      4      dcursoraddr:=frameaddr;
473:C      4      areaisdbcr:=true;
474:C      4      end;
475:C
476:C      4      dbexcg:
477:C      4      begin
478:C      4      changecursor;
479:C      4      cexchange( savearea, ymin, ymax, xmin, (xmax-xmin+1)*8);
480:C

```



```

481:C      4      tempaddr:=cursoraddr;
482:C      4      cursoraddr:=dcursoraddr;
483:C      4      dcursoraddr:=tempaddr;
484:C      4      changecursor;
485:C      4      areaisdbcr:=not areaisdbcr;
486:C      4      end;
487:S
488:C      4
489:C      4      end; { of case }
490:C      3      end; { of with }
491:C      2      end; { crtdebug procedure }
492:S
493:D      1      procedure dummy;
494:C      2      begin end;
495:S
496:S
497:D      1      procedure gatorcrtinit;
498:C      -2      var i: shortint;
499:S
500:D      2      begin
501:C      2      id:=245; { set io char to roman8 value }
502:C      2      with syscom^.crtinfo do begin
503:C      3      screenwidth:=width;
504:C      3      screenheight:=height;
505:C      3      maxx:=screenwidth-1;
506:C      3      maxy:=screenheight-1;
507:C      3      screensize:=screenwidth*screenheight;
508:C      3      cbuidtable;
509:C      3      highlight:=0; defaulthighlight:=0;
510:C      3      gotoxy(0,0);
511:C      3      dumpalphahook := dumpg;
512:C      3      dumpgraphicshook := dumpg;
513:C      3      updatecursorhook:=doupdatecursor;
514:C      3      dbcrthook:=crtdebug;
515:C      3      crtllhook:=lineops;
516:C      3      crtiohook:=docrtio;
517:C      3      crtinithook:=gatorcrtinit;
518:C      3      togglealphahook:=dummy;
519:C      3      togglegraphicshook:=dummy;
520:C      3      currentcrt:=bitmaptype;
521:C      3      keybuffer^.maxsize:=maxx-8;
522:C      3      end;
523:C      2      end;
524:D      -18      1
525:S
526:S
527:D      1      function gatorcrttype:boolean;
528:C      2      const gatorid=25;
529:D      -4      var ptr: shortint;
530:C      -6      i: shortint;
531:D      -8      dummy: shortint;
532:D      -9      found: boolean;
533:S
534:S
535:C      2      begin
536:C      2      bitmapaddr:=0;
537:C      2      found:=false;
538:C      2      ptr:=anyptr(hex('560000'));
539:C      2      try
540:C      2      try

```

```

541:C      3      dummy:=ptr^;
542:C      3      if (dummy mod 128) = gatorid then begin
543:C      4      found:=true; bitmapaddr:=integer(ptr);
544:C      4      end;
545:C      3      recover
546:C      3      if escapecode<>-12 then escape(escapecode);
547:S
548:S
549:C      2      {***** NOTE COMMENTED OUT CODE BELOW *****}
550:C      {ptr:=anyptr(hex('680000'))};
551:C      i:=0;
552:C      while (i<=31) and not found do begin
553:C      try
554:C      dummy:=ptr^;
555:C      if (dummy mod 128) = gatorid then begin
556:C      found:=true; bitmapaddr:=integer(ptr);
557:C      end;
558:C      recover
559:C      if escapecode<>-12 then escape(escapecode);
560:S
561:C      ptr:=anyptr(integer(ptr)+65536);
562:C      i:=i+1;
563:C      2      end; }
564:S
565:C      2      gatorcrttype:=found;
566:S
567:C      2      if found then begin
568:C      3      syscom^.environc;
569:C      3      gatorcrtinit;
570:C      3      end;
571:S
572:C      2      end;
573:S
574:S
575:S
576:S
577:S
578:C      1      end; { of module -- I hope }
579:S
580:D      1      import crt, loader;
581:S
582:C      1      begin
583:C      1      if gatorcrttype then markuser;
584:C      1      end.
585:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

G_DRV

Description

G_DRV provides low-level driver support.

Usage

G_DRV is used by the 98622 interface.

Requirements

GPIO and the I/O library kernel (IODECLARATIONS, etc.)

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*)
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1983.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S
11:S      RESTRICTED RIGHTS LEGEND
12:S      Use, duplication, or disclosure by the Government
13:S      is subject to restrictions as set forth in
14:S      paragraph (b) (3) (B) of the Rights in Technical
15:S      Data and Computer Software clause in
16:S      DAR 7-104.9(a).
17:S
18:S      HEWLETT-PACKARD COMPANY
19:D      0 Fort Collins, Colorado      *)
20:S
21:S
22:D      0 $SEARCH 'IOLIB:KERNEL.CODE'$
23:D      0 $MODCAL ON$
24:D      0 $PARTIAL_EVAL ON$
25:D      0 $STACKCHECK ON$
26:D      0 $RANGE OFF$
27:D      0 $DEBUG OFF$
28:D      0 $OVFLCHECK OFF$

```

```

29:D      0 $PAGES$
30:S
31:D      0 *****
32:D      0 *
33:D      0 *      not RELEASED      VERSION      3.0      *)
34:D      0 *
35:D      0 *****
36:D      0 *
37:D      0 *
38:D      0 *      IOLIB      GPIO_DRIVERS
39:D      0 *
40:D      0 *
41:D      0 *****
42:D      0 *
43:D      0 *
44:D      0 *      library      - IOLIB
45:D      0 *      name      - GPIO_DRIVERS
46:D      0 *      module(s) - init_gpio
47:D      0 *      extg
48:D      0 *
49:D      0 *      author
50:D      0 *      phone
51:D      0 *
52:D      0 *      date      - June 1 , 1981
53:D      0 *      update     - Aug 1 , 1983
54:D      0 *      release    - ?????????????
55:D      0 *
56:D      0 *      source     - IOLIB:G_DRV.TEXT
57:D      0 *      object     - IOLIB:G_DRV.CODE
58:D      0 *
59:D      0 *****

```



```

60:D 0 $PAGE$
61:D 0 (*****
62:D 0 *
63:D 0 *
64:D 0 *      BUG FIX HISTORY      - after release 1.0
65:D 0 *
66:D 0 *
67:D 0 *      BUG #    BY    / ON      LOC      DESCRIPTION
68:D 0 *      -----
69:D 0 *      367      -----      gpio_initialize Allow eXecute of driver
70:D 0 *      09/22/82      and have it install
71:D 0 *      itself in the system.
72:D 0 *
73:D 0 *
74:D 0 (*****

```

```

75:D 0 $PAGE$
76:D 0 (*****
77:D 0 *
78:D 0 *
79:D 0 *      This is the source code for an external procedures library
80:D 0 *      to be used for general purpose interfacing on the HP 9826.
81:D 0 *
82:D 0 *      The library consists of 3 primary sets of modules -
83:D 0 *
84:D 0 *          1.      KERNEL modules
85:D 0 *          2.      driver modules
86:D 0 *          3.      IOLIB modules
87:D 0 *
88:D 0 *      The KERNEL modules consist of the following modules -
89:D 0 *
90:D 0 *          1.      iodeclarations  ( contains static r/w space )
91:D 0 *          2.      iocomasm
92:D 0 *          3.      general_0      ( initialization & low level
93:D 0 *                               routines like ioread/iowrite)
94:D 0 *
95:D 0 *      The KERNEL modules also have an executable program segment
96:D 0 *      that gets executed at the time it is loaded. This program
97:D 0 *      initializes the static read/write memory. This program also
98:D 0 *      allocates the temporary storage for any card that exists -
99:D 0 *      independent of whether there is or is not a driver for it.
100:D 0 *
101:D 0 *      The driver modules consist of the actual assembly or PASCAL
102:D 0 *      routines that deal with a specific interface card. There is
103:D 0 *      also an executable program segment for each driver module.
104:D 0 *
105:D 0 *      This program searches the select code table in the static r/w
106:D 0 *      initialized by the KERNEL general_0 module for all select codes
107:D 0 *      that have the right interface card ( HP1B drivers will search
108:D 0 *      for the 98624 interface ). This program will then set up the
109:D 0 *      driver tables to point to the correct drivers.
110:D 0 *
111:D 0 *      The rest of the IOLIB modules are high-level modules that are
112:D 0 *      used by an end user in his/her application program.
113:D 0 *
114:D 0 *      The KERNEL and some set of driver modules will exist in the
115:D 0 *      SYSTEM.INITLIB file as object code ( not EXPORT text ). The
116:D 0 *      export text will reside on the SYSTEM.LIBRARY file. The rest
117:D 0 *      of the library will reside on the SYSTEM.LIBRARY.

```



```
118:D      0 $PAGE$
119:D      0 (*****
120:D      0 *
121:D      0 *
122:D      0 *      REFERENCES :
123:D      0 *
124:D      0 *
125:D      0 *      1. 9826 I/O Designers Guide      ( ----- )
126:D      0 *
127:D      0 *      2. 68000 Manual      ( Motorola )
128:D      0 *
129:D      0 *      3. Pascal alpha site ERS      ( ----- )
130:D      0 *
131:D      0 *      4. Pascal I/O Library ERS      ( ----- )
132:D      0 *
133:D      0 *      5. 9826 HPL EIO & IOD listings      ( ----- )
134:D      0 *
135:D      0 *      6. 9826 HPL Misc. I/O Doc.      ( ----- )
136:D      0 *
137:D      0 *      7. 9826 card documentation      ( Mfg. Specs. )
138:D      0 *
139:D      0 *      8. Pascal I/O Library IRS      ( ----- )
140:D      0 *
141:D      0 *
142:D      0 (*****
```

```
143:D      0 $PAGE$
144:D      0 PROGRAM gpio_initialize ( INPUT , OUTPUT );
```



```

145:D 1 $PAGES
146:D 1 {*****}
147:D 1 {*}
148:D 1 {*}
149:D 1 {*      GPIO DRIVERS}
150:D 1 {*}
151:D 1 {*}
152:D 1 {*****}
153:D 1 EXTERNAL MODULE extg;
154:S {
155:S   { date    08/25/81
156:S     update 10/13/81
157:S   }
158:S   purpose This module is a declaration of the importation text for
159:S     the external drivers.
160:S   }
161:S   note   The assembly language code that is imported needs to be
162:S     called 'extg'. The routines need to be called
163:S     'extg_@0000' - eg_init referenced below would be
164:S     extg_eg_init. 'eg' refers to GPIO.
165:D 1   }
166:S
167:D 1 IMPORT  sysglobals ,
168:D 1         iodeclarations ;
169:S
170:D 1 EXPORT
171:S
172:D 1 PROCEDURE eg_init ( temp : ANYPTR );
173:D 1 PROCEDURE eg_isr  ( temp : PISRI8 );
174:D 1 PROCEDURE eg_rdb  ( temp : ANYPTR ; VAR x : CHAR);
175:D 1 PROCEDURE eg_wtb  ( temp : ANYPTR ; val : CHAR);
176:D 1 PROCEDURE eg_rdw  ( temp : ANYPTR ; VAR x : io_word);
177:D 1 PROCEDURE eg_tdw  ( temp : ANYPTR ; val : io_word);
178:D 1 PROCEDURE eg_rds  ( temp : ANYPTR ; reg : io_word;
179:D 2         VAR x : io_word);
180:D 1 PROCEDURE eg_wtc  ( temp : ANYPTR ; reg : io_word;
181:D 2         val : io_word );
182:D 1 PROCEDURE eg_tfr  ( temp : ANYPTR ; bcb : ANYPTR );
183:D 1 PROCEDURE eg_clr  ( temp : ANYPTR ; line : io_bit );
184:D 1 PROCEDURE eg_set  ( temp : ANYPTR ; line : io_bit );
185:D 1 PROCEDURE eg_test ( temp : ANYPTR ; line : io_bit ;
186:D 2         VAR x : BOOLEAN );
187:S
188:D 1 END; { of extg }

```

```

189:D 1 $PAGES
190:S
191:S 1 MODULE init_gpio;
192:D 1 {
193:S   { date    08/25/81
194:S     update 10/04/82
195:S     8/01/83   change rev number
196:S   }
197:S   purpose This module initializes the gpio drivers.
198:S   }
199:D 1 }
200:D
201:S
202:D 1 IMPORT  iodeclarations ;
203:D
204:S
205:D 1 EXPORT
206:S
207:S
208:D 1 VAR
209:D -120 1   gpio_drivers : drv_table_type;
210:D
211:D -120 1 PROCEDURE io_init_gpio;
212:S
213:S
214:D -120 1 IMPLEMENT
215:D
216:D
217:D
218:D -120 1 IMPORT  sysglobals ,
219:D -120 1         isr ,
220:D -120 1         genefal_0 ,
221:D -120 1         extg ;
222:S
223:S
224:D 1 PROCEDURE io_init_gpio;
225:D 2 VAR
226:D -2 2   io_isc      : type_isc;
227:D -6 2   dummy      : INTEGER;
228:D -8 2   io_lvl     : io_byte;
229:C 2 BEGIN
230:S
231:C 2   io_revcd := io_revcd + ' G3.0';      ( GPIO revision added 2/5/82 TM )
232:C
233:C 2 { set up the driver tables }
234:S
235:C 2 WITH gpio_drivers DO BEGIN
236:C 3   gpio_drivers := dummy_drivers ;
237:C 3   iod_init := eg_init;
238:C 3   iod_isr  := eg_isr;
239:C 3   iod_rdb  := eg_rdb;
240:C 3   iod_wtb  := eg_wtb;
241:C 3   iod_rdw  := eg_rdw;
242:C 3   iod_tdw  := eg_tdw;
243:C 3   iod_rds  := eg_rds;
244:C 3   iod_wtc  := eg_wtc;
245:C 3   iod_tfr  := eg_tfr;
246:C 3   iod_set  := eg_set;
247:C 3   iod_clr  := eg_clr;
248:C 3   iod_test := eg_test;

```



```

249:C      3      END; { of WITH }
250:S
251:S
252:C      2      { set up drivers for the interfaces }
253:S
254:C      2      FOR io_isc:=iomisc TO iomaxisc DO
255:C      3      WITH isc_table[io_isc] DO BEGIN
256:S
257:C      4          IF card_id = hp98622
258:C      5          THEN BEGIN
259:S
260:C      5              io_drv_ptr:=ADDR(gpio_drivers),
261:C      5              WITH io_drv_ptr^, io_tmp_ptr^ DO BEGIN
262:S
263:C      6              { if the card exists then link in an ISR for it }
264:C      6              { ??? - what happens if an ISR fires during init }
265:S
266:C      6              io_lvl:=(io_read_byte(io_isc,3) DIV 16) MOD 4)+3;
267:C      6              IF myisrib.INTEGADDR <>~NIL
268:C      7              THEN BEGIN
269:C      7                  { if isr exists then unlink it }
270:C      7                  ISRUNLINK(io_lvl,
271:C      7                      addr(myisrib));      { interrupt level      }
272:C      7                      { ptr to the isrib      }
273:C      7                  END; { of IF }
274:S
275:C      6              PERMISRLINK(io_drv_ptr^,
276:C      6                      ANYPTR(INTEGER(card_ptr)+3),
277:C      6                      192,
278:C      6                      192,
279:C      6                      io_lvl,
280:C      6                      ADDR(myisrib));
281:C      6                      { isr
282:C      6                      { int reg addr }
283:C      6                      { int reg mask }
284:C      6                      { int reg value }
285:C      6                      { int level }
286:C      6                      { isrib info }
287:S
288:C      6              END; { of WITH BEGIN }
289:S
290:C      5              END. { of IF card_type = gpio_card }
291:S
292:C      4              END; { of FOR io_isc WITH isc_table[io_isc] BEGIN }
293:S
294:C      2      { call the actual driver initialization }
295:S
296:C      2      FOR io_isc:=iomisc TO iomaxisc DO
297:C      3      WITH isc_table[io_isc] DO
298:C      4      IF card_id = hp98622
299:C      5      THEN BEGIN
300:C      5      CALL(io_drv_ptr^.iod_init , io_tmp_ptr);
301:C      5      END; { of WITH IF }
302:S
303:C      2      END; { of io_init_gpio }
304:S
305:C      1      END; { of MODULE init_gpio }

```

```

302:D      1      $PAGE$
303:S
304:D      1      IMPORT      init_gpio ,
305:D      1      LOADER ;
306:S
307:C      1      BEGIN
308:C      1      io_init_gpio;
309:C      1      MARKUSER;
310:C      1      END. { of gpio_initialize }
311:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

Description

GEN contains most of the viewing transformations and color model transformations for DGL.

Requirements

DGL_TYPES, ASM, DGL_VARS, GLE_TYPES, and GLE_GEN.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:D 0 {
2:D 0 { Pascal work station graphics library }
3:D 0 {
4:D 0 { Module = DGL_GEN }
5:D 0 { Programmer = BJS }
6:D 0 { Date = 2/1/81 }
7:S }
8:D 0 { Purpose: To hold most internal routines }
9:S }
10:D 0 { Rev history }
11:D 0 { 6-15-82 BJS - Added moonunit dump graphics stuff }
12:D 0 { 7-05-82 BJS - Removed HPGL clipping, now uses ASM_RAS clipping }
13:D 0 { 7-05-82 BJS - Changes to add 9836C proto support }
14:D 0 { 8-25-82 BJS - Major mods for GLE }
15:D }
16:D { (c) Copyright Hewlett-Packard Company, 1983.
17:D All rights are reserved. Copying or other
18:D reproduction of this program except for archival
19:D purposes is prohibited without the prior
20:D written consent of Hewlett-Packard Company.
21:D }
22:D
23:D RESTRICTED RIGHTS LEGEND
24:D
25:D Use, duplication, or disclosure by the Government
26:D is subject to restrictions as set forth in
27:D paragraph (b) (3) (B) of the Rights in Technical
28:D Data and Computer Software clause in
29:D DAR 7-104.9(a).
30:D
31:D HEWLETT-PACKARD COMPANY
32:D Fort Collins, Colorado }
33:D
34:D 0 $mcdcal$
35:D 0 $ref 50$
36:S
37:D 0 $include 'OPTIONS'$
38:D { This include file specifies range checking, debug and other compiler
39:D options for the graphics library }
40:S
41:D 0 $debug OFF$
42:D 0 $range OFF$
43:D 0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
44:D 0 $FLOAT_HOW TESTS
45:S
46:D
47:S
48:D
49:D 0 $include 'OPTIONS'$
3000:D 0 $linenum 3000$
3001:S
3002:D 0 $search 'TYPES'
3003:D 0 'DGL_VARS',
3004:D 0 'GLE_LIB'$
3005:S
3006:D 0 module DGL_GEN;
3007:S
3008:D 1 import DGL_TYPES;
3009:S

```

```

3010:D 1 export
3011:S
3012:D 1 procedure error ( error_number : integer );
3013:D
3014:D 1 function min(p1,p2:integer) : integer;
3015:S
3016:D 1 function max(p1,p2:integer) : integer;
3017:S
3018:D 1 procedure locator_limits (lxmin, lxmax,
3019:D 2 lymin, lymax : real);
3020:S
3021:D 1 procedure calculate_viewing;
3022:S
3023:D 1 procedure display_limits ( dxmin, dxmax,
3024:D 2 dymin, dymax : real);
3025:S
3026:D 1 procedure convert_intwtod (x, y : gshortint;
3027:D 2 var dx, dy : integer );
3028:S
3029:D 1 procedure convert_wtod (x, y : real;
3030:D 2 var dx, dy : integer );
3031:S
3032:D 1 procedure convert_ltod (lx, ly : integer;
3033:D 2 var dx, dy : integer );
3034:S
3035:D 1 procedure convert_dtow (dx, dy : integer;
3036:D 2 var wx, wy : real);
3037:S
3038:D 1 procedure ck_system_init;
3039:S
3040:D 1 procedure ck_display_init;
3041:S
3042:D 1 procedure ck_locator_init;
3043:S
3044:D 1 procedure adjust_return_echo ( var rx,ry : real );
3045:S
3046:D 1 function opcode_ck(opcode,num_integer,num_real : integer) : integer;
3047:S
3048:D 1 procedure convert_hsl_to_rgb ( hue,sat,life : real; var r,g,b : real);
3049:D 1 procedure convert_rgb_to_hsl (r,g,b : real; var hue,sat,life : real);
3050:S
3051:D 1 implement
3052:S
3053:D 1 import asm,
3054:D 1 dgl_vars,
3055:D 1 gle_gen,
3056:D 1 GLE_TYPES;
3057:S
3058:D 1 { *****
3059:D 1 {
3060:D 1 { General procedures and functions
3061:D 1 { *****
3062:D 1 {
3063:S
3064:D 1 procedure error ( error_number : integer);
3065:S
3066:D 2 { Purpose : To log a graphics error, and perform a graphics error escape }
3067:S
3068:D 2 begin
3069:D 2 graphics_error := error_number;

```



```

3070:C      2  escape(graphics_error_number);
3071:C      2  end; { error }
3072:S
3073:D      1  procedure ck_system_init;
3074:S
3075:D      2  { Purpose : To report an error if the system is not initialized      }
3076:S
3077:C      2  begin
3078:C      2  if not system_init then error (err_sys_int);
3079:C      2  end; { ck_sytem_init }
3080:S
3081:D      1  procedure ck_display_init;
3082:S
3083:D      2  { Purpose : To report an error if the display is not initialized      }
3084:S
3085:C      2  begin
3086:C      2  if not disp_init then error (err_dis_int);
3087:C      2  end; { ck_display_init }
3088:S
3089:D      1  procedure ck_locator_init;
3090:S
3091:D      2  { Purpose : To report an error if the locator is not initialized      }
3092:S
3093:C      2  begin
3094:C      2  if not loc_init then error (err_loc_int);
3095:C      2  end; { ck_locator_init }
3096:S
3097:D      1  function min(p1,p2:integer):integer;
3098:S
3099:D      2  { Purpose : To return the minimum of two integers      }
3100:S
3101:C      2  begin
3102:C      2  if (p1 >= p2) then min := p2
3103:C      3  else min := p1;
3104:C      2  end; { min }
3105:S
3106:D      1  function max(p1,p2:integer):integer;
3107:S
3108:D      2  { Purpose : To return the maximum of two integers      }
3109:S
3110:C      2  begin
3111:C      2  if (p1 <= p2) then max := p2
3112:C      3  else max := p1;
3113:C      2  end; { max }
3114:C
3115:S
3116:D      1  procedure order_values (      p1,p2 : real;
-16 2      var rp1, rp2 : real);
3117:D
3118:S
3119:D      2  { Purpose : To return the passed in values ordered such that the smaller
-16 2      value is in rp1.      }
3120:D
3121:S
3122:C      2  begin
3123:C      2  if p1 <= p2 then
3124:C      3  begin
3125:C      3  rp1 := p1;
3126:C      3  rp2 := p2;
3127:C      3  end
3128:C      3  else
3129:C      3  begin

```

```

3130:C      3  rp2 := p1;
3131:C      3  rp1 := p2;
3132:C      3  end;
3133:C      2  end; { order_values }
3134:S
3135:D      1  function opcode_ck(opcode,num_integer,num_real : integer) : integer;
3136:S
3137:D      2  { Purpose: To ck for a valid opcode passed. Opcodes are all defined such      }
3138:D      { that:
3139:D      the hundreds digit = # of reals passed
3140:D      the 1000's digit = # of integers passed
3141:D      This function returns values as follows:
3142:D      = 0 ; ok
3143:D      = 2 ; wrong i size
3144:D      = 3 ; wrong r size
3145:D      }
3146:D
3147:D      2
3148:S
3149:S
3150:D      2  var
-10 2      digit : array [0..4] of gbyte;
-12 2      i : gshortint;
3153:S
3154:C      2  begin
3155:C      2  for i := 0 to 4 do
3156:C      3  begin
3157:C      3  digit[i] := opcode - (opcode div 10) * 10;
3158:C      3  opcode := opcode div 10;
3159:C      3  end;
3160:S
3161:C      2  opcode_ck := 0;
3162:C      2  if digit[2] <> num_real then opcode_ck := 3;
3163:C      2  if digit[3] <> num_integer then opcode_ck := 2;
3164:C      2  end;
3165:S
3166:D      -24 1  procedure convert_hsl_to_rgb ( hue,sat,lite : real; var r,g,b : real);
3167:S
3168:D      -24 2  var
-64 2      lx,ly,lz,frac,h : real;
-68 2      j : integer;
3171:S
3172:C      2  begin
3173:C      2  h := 6*hue;
3174:C      2  i := trunc(h);
3175:C      2  frac := h-i;
3176:C      2  lx := lite*(1.0-sat);
3177:C      2  ly := lite*(1.0-sat*frac);
3178:C      2  lz := lite*(1.0-sat*(1-frac));
3179:S
3180:C      2  case j of
3181:C      3  0,6 : begin r := lite; g := lz; b := lx; end;
3182:C      3  1 : begin r := ly; g := lite; b := lx; end;
3183:C      3  2 : begin r := lx; g := lite; b := lz; end;
3184:C      3  3 : begin r := lx; g := ly; b := lite; end;
3185:C      3  4 : begin r := lz; g := lx; b := lite; end;
3186:C      3  5 : begin r := lite; g := lx; b := ly; end;
3187:C      2  end; { of case }
3188:C      2  end;
3189:S

```



```

3190:D -24 1 procedure convert_rgb_to_hsl (r,g,b : real; var hue,sat,lite : real);
3191:S
3192:D -32 2   var x : real;
3193:D -56 2   tr,rg,tb : real;
3194:S
3195:D -24 2   function max3(a,b,c : real) : real;
3196:S
3197:C 3   begin
3198:C 4     if (a >= b) and (a >= c) then max3 := a
3199:C 4     else
3200:C 4       if (b >= a) and (b >= c) then max3 := b
3201:C 4       else
3202:C 5         max3 := c;
3203:C 3     end;
3204:S
3205:D -24 2   function min3(a,b,c : real) : real;
3206:S
3207:C 3   begin
3208:C 4     if (a <= b) and (a <= c) then min3 := a
3209:C 4     else
3210:C 4       if (b <= a) and (b <= c) then min3 := b
3211:C 4       else
3212:C 5         min3 := c;
3213:C 3     end;
3214:S
3215:C 2 begin
3216:C 2   hue := 0;           ( init values given, since values may be undefined )
3217:C 2   sat := 1;
3218:S
3219:C 2   lite := max3(r,g,b);
3220:C 2   x := min3(r,g,b);
3221:C 2   if lite <> 0 then ( calc hue and sat only if defined )
3222:C 3   begin
3223:C 3     sat := (lite - x) / lite;
3224:C 3     if sat <> 0 then ( calc hue only if defined, other wise defaults to old )
3225:C 4     begin
3226:C 4       tr := (lite-r)/(lite-x);
3227:C 4       rg := (lite-g)/(lite-x);
3228:C 4       tb := (lite-b)/(lite-x);
3229:C 4
3230:S
3231:C 4       if r=lite then
3232:C 5       begin
3233:C 5         if g=x then hue := 5+tb;
3234:C 5         else hue := 1-tg;
3235:C 5       end
3236:C 5       else
3237:C 5       if g=lite then
3238:C 6       begin
3239:C 6         if b=x then hue := 1+tr;
3240:C 6         else hue := 3-tb;
3241:C 6       end
3242:C 6       else
3243:C 6       begin
3244:C 6         if r=x then hue := 3+tg;
3245:C 6         else hue := 5-tr;
3246:C 6       end;
3247:C 4     hue := hue/6;
3248:C 4     end;
3249:C 3   end;

```

```

3250:C 2 end
3251:S
3252:D 1 procedure adjust_return_echo ( var rx,ry : real );
3253:S
3254:D 2 { Purpose : wlast return echo values for the effects of rubber band snap }
3255:S
3256:C 2 begin
3257:C 2   case current_echo_type of
3258:C 3     5: ry := w_loc_echo_y;   { Horizontal rubber band }
3259:C 3     6: rx := w_loc_echo_x;   { Vertical rubber band }
3260:C 3     7: { snap horiz / vert rubber band }
3261:C 3     if abs(rx-w_loc_echo_x) >= abs(ry-w_loc_echo_y) then
3262:C 4       ry := w_loc_echo_y;
3263:C 4     else rx := w_loc_echo_x; { no adjustment needed for others }
3264:C 3   otherwise ;
3265:C 3 end; { of case }
3266:C 2 end; { adjust_return_echo }
3267:S
3268:D 1 { ***** }
3269:D 1 {
3270:D 1   General Viewing transformation routines
3271:D 1 {
3272:D 1 { ***** }
3273:S
3274:D 1 procedure convert_intwtod( x,y: gshortint;
3275:D 2   var dx,dy: integer);
3276:S
3277:D 2 { Purpose : To convert form world to display cord }
3278:S
3279:D 2 var
3280:D -8 2   tx, ty : integer;
3281:D -12 2   lx,ly : gshortint;
3282:S
3283:C 2 begin
3284:C 2   if short_defaults then
3285:C 3   begin
3286:C 3     dx := x;
3287:C 3     dy := y;
3288:C 3   end
3289:C 3   else
3290:C 3     with scalef do
3291:C 4     begin
3292:C 4       tx := x * x_display_delta;
3293:C 4       lx := tx div x_window_delta + x_display_offset;
3294:C 4       ty := y * y_display_delta;
3295:C 4       ly := ty div y_window_delta + y_display_offset;
3296:C 4       dx := lx;
3297:C 4       dy := ly;
3298:C 4     end;
3299:S
3300:C 2 end; { convert_intwtod }
3301:S
3302:D 1 procedure convert_wtod( x,y:real;
3303:D 2   var dx,dy:integer);
3304:S
3305:D -16 2 { Purpose : To convert form world to display cord }
3306:S
3307:C 2 begin
3308:C 2   dx := trunc(x * xwtod_scale + xwtod_offset);
3309:C 2   dy := trunc(y * ywtod_scale + ywtod_offset);

```



```

3310:C      2  {WRITELN('WTOD ('X:8:S,Y:8:S,')='DX:8,DY:8);}
3311:C      2  end; { convert_wtod }
3312:S
3313:D      1  procedure convert_ltod( lx,ly : integer;
3314:D      2  var dx,dy : integer);
3315:S
3316:D      2  { Purpose : To convert from locator to display units }
3317:S
3318:C      2  begin
3319:C      3  with gcb^ do
3320:C      3  begin
3321:C      3  dx := trunc( ((lx - log_loc_lim.xmin) * xltod_scale) +
3322:C      3  cur_disp_lim.xmin + 0.5);
3323:C      3  dy := trunc( ((ly - log_loc_lim.ymin) * yltod_scale) +
3324:C      3  cur_disp_lim.ymin + 0.5);
3325:C      3  end;
3326:C      2  end; { convert_ltod }
3327:S
3328:D      1  procedure convert_dtow( dx,dy : integer;
3329:D      2  var wx,wy : real);
3330:S
3331:D      2  { Purpose : To convert from display to world units }
3332:S
3333:C      2  begin
3334:C      2  wx := (dx + xdtow_offset) * xdtow_scale;
3335:C      2  wy := (dy + ydtow_offset) * ydtow_scale;
3336:C      2  end; { convert_dtow }
3337:S
3338:S
3339:S
3340:D      1  procedure locator_limits (lxmin,lxmax,
3341:D      2  lymin,lymax : real);
3342:S
3343:D      -32 2 { Purpose : To set the locator limits ( in device units ) }
3344:D      -32 2 { Note: This routine does not perform error cking }
3345:S
3346:C      2  begin
3347:C      3  with gcb^ do
3348:C      3  begin
3349:C      3  { set the logical limits }
3350:C      3  with log_loc_lim do
3351:C      3  begin
3352:C      4  xmin := lxmin;
3353:C      4  xmax := lxmax;
3354:C      4  ymin := lymin;
3355:C      4  ymax := lymax;
3356:C      4  end;
3357:S
3358:C      3  { calculate the display / locator xforms }
3359:S
3360:C      3  xltod_scale := (cur_disp_lim.xmax - cur_disp_lim.xmin) /
3361:C      3  (log_loc_lim.xmax - log_loc_lim.xmin);
3362:S
3363:C      3  yltod_scale := (cur_disp_lim.ymax - cur_disp_lim.ymin) /
3364:C      3  (log_loc_lim.ymax - log_loc_lim.ymin);
3365:S
3366:S
3367:C      3  { set the locator echo position to the center of the window }
3368:S
3369:C      3  with window_lim do

```

```

3370:C      4  begin
3371:C      4  w_loc_echo_x := ((xmax - xmin) / 2.0) + xmin;
3372:C      4  w_loc_echo_y := ((ymax - ymin) / 2.0) + ymin;
3373:C      4  end;
3374:S
3375:C      3  { convert to device units }
3376:S
3377:C      3  convert_wtod ( w_loc_echo_x, w_loc_echo_y,
3378:C      3  d_loc_echo_x, d_loc_echo_y );
3379:S
3380:C      3  end;
3381:S
3382:C      2  end; { locator_limits }
3383:S
3384:S
3385:D      1  procedure calculate_viewing;
3386:S
3387:D      2  { Purpose : To calculate a new viewing transformation }
3388:S
3389:D      2  var
3390:D      -8  temp : real; { temp work var }
3391:S
3392:D      2  procedure int_test;
3393:S
3394:D      3  { Purpose : To see if conditions are right, and setup if so, for int_move
3395:D      3  and int_line to call fast integer device routines }
3396:S
3397:D      3  var
3398:D      -4  3  sxmin : integer;
3399:D      -8  3  sxmax : integer;
3400:D      -12 3  symin : integer;
3401:D      -16 3  symax : integer;
3402:D      -18 3  gsxmin : gshortint;
3403:D      -20 3  gsxmax : gshortint;
3404:D      -22 3  gsymin : gshortint;
3405:D      -24 3  gsymin : gshortint;
3406:S
3407:D      -28 3  window_delta_x : integer;
3408:D      -32 3  window_delta_y : integer;
3409:S
3410:C      3  begin
3411:C      3  short_flag := false;
3412:S
3413:C      3  with gcb^ do
3414:C      4  begin
3415:C      4  if disp_init then
3416:C      5  try
3417:C      6  $range on$ { range cking done here }
3418:S
3419:C      6  { find device points for the corners of the window }
3420:C      6  convert_wtod(window_lim.xmin,window_lim.ymin,sxmin,symin);
3421:C      6  convert_wtod(window_lim.xmax,window_lim.ymax,sxmax,symax);
3422:S
3423:C      6  { convert to non-fix point form }
3424:C      6  gsxmin := sxmin;
3425:C      6  gsxmax := sxmax;
3426:C      6  gsymin := symin;
3427:C      6  gsymin := symin;
3428:S
3429:C      6  { set up scale int-

```



```

3430:C      6      with window_lim do
3431:C      7      begin
3432:C      7      window_delta_x := trunc(xmax) - trunc(xmin);
3433:C      7      window_delta_y := trunc(ymax) - trunc(ymin);
3434:S      7
3435:C      7      { ck delta window values in range }
3436:C      7      if (abs(window_delta_x) > 32767) or
3437:C      8      (abs(window_delta_y) > 32767) then
3438:C      8      escape(-4); { force integer overflow }
3439:S      8
3440:C      7      with scalef do
3441:C      8      begin
3442:C      8      x_display_delta := gsxmax - gsxmin;
3443:C      8      x_window_delta := window_delta_x;
3444:C      8      x_display_offset :=
3445:C      8      ((-trunc(xmin) * x_display_delta) div x_window_delta) + gsxmin;
3446:C      8      y_display_delta := gsymax - gsymin;
3447:C      8      y_window_delta := window_delta_y;
3448:C      8      y_display_offset :=
3449:C      8      ((-trunc(ymin) * y_display_delta) div y_window_delta) + gsymin;
3450:S      8
3451:S      8
3452:C      8      if (x_display_offset = 0) and
3453:C      9      (x_window_delta = x_display_delta) and
3454:C      9      (y_display_offset = 0) and
3455:C      9      (y_window_delta = y_display_delta) then
3456:C      9      short_defaults := true
3457:C      9      else
3458:C      9      short_defaults := false;
3459:S      9
3460:C      8      end;
3461:C      7      end;
3462:C      6      short_flag := true;
3463:S      6
3464:C      6      recover { can't use int_move / int_line internal routines }
3465:C      6      if (escapecode > -4) or { ignore all math and range errors }
3466:C      7      (escapecode < -8) then escape(escapecode);
3467:C      4 $range off$
3468:C      4      end;
3469:C      3      end; { of procedure int_test }
3470:S      3
3471:S      2
3472:C      2 begin
3473:C      2 with gcb^ do
3474:C      3 begin
3475:C      3 { set view surface limits to the logical display limits }
3476:C      3 cur_disp_lim := log_disp_lim;
3477:S      3
3478:C      3 { if desired aspect = aspect of logical limits then we don't need to }
3479:C      3 redefine the limits as set above
3480:C      3 if (aspect_ratio <> log_aspect) then
3481:C      4 begin
3482:C      4 {determine which device limits to change}
3483:C      4 if (aspect_ratio <= log_aspect) then
3484:C      5 begin
3485:C      5 {check display justification}
3486:C      5 if (disp_just = centered) then
3487:C      5
3488:C      5
3489:S      5

```

```

3490:S      6      {set vertical limits for device on which view space is
3491:C      6      centered within the logical display limits }
3492:C      6      begin
3493:C      6      temp := 0.5 * (dyunits - (dxunits * aspect_ratio));
3494:C      6      cur_disp_lim.ymin := log_disp_lim.ymin + temp;
3495:C      6      cur_disp_lim.ymax := log_disp_lim.ymax - temp;
3496:C      6      end
3497:S      6
3498:C      6      else { not centered }
3499:S      6
3500:C      6      {set vertical limits for device on which view space is
3501:C      6      lower left justified within the logical display limits }
3502:C      6      begin
3503:C      6      cur_disp_lim.ymax :=
3504:C      6      log_disp_lim.ymin + (dxunits * aspect_ratio);
3505:C      6      end;
3506:C      6      end
3507:S      6
3508:C      5      else { current aspect ratio > aspect ratio of logical limits }
3509:C      5      begin
3510:S      5
3511:C      5      {check display justification}
3512:C      5      if (disp_just = centered) then
3513:C      6      {set horizontal limits for device on which view space is
3514:C      6      centered within the logical display limits }
3515:C      6      begin
3516:C      6      temp := 0.5 * (dxunits - (dyunits / aspect_ratio));
3517:C      6      cur_disp_lim.xmin := log_disp_lim.xmin + temp;
3518:C      6      cur_disp_lim.xmax := log_disp_lim.xmax - temp;
3519:C      6      end
3520:C      6      end
3521:S      6
3522:C      6      else { not centered }
3523:S      6
3524:C      6      {set horizontal limits for device on which view space is
3525:C      6      lower left justified within the logical display limits }
3526:C      6      begin
3527:C      6      cur_disp_lim.xmax :=
3528:C      6      log_disp_lim.xmin + (dyunits / aspect_ratio);
3529:C      6      end;
3530:C      6      end;
3531:C      4      end;
3532:S      4
3533:S      3
3534:C      3      { set clipping limits here }
3535:C      3      if disp_init then
3536:C      4      with gle_gcb^, cur_disp_lim do
3537:C      5      begin
3538:C      5      (WRITELN('C = ', XMIN:19:18, XMAX:19:18, YMIN:19:18, YMAX:19:18);
3539:C      5      WRITELN('CI = ', TRUNC(XMIN+0.5):19, TRUNC(XMAX+0.5):19,
3540:C      5      TRUNC(YMIN+0.5):19, TRUNC(YMAX+0.5):19));
3541:C      5      info1 := trunc(xmin+0.5);
3542:C      5      info2 := trunc(xmax+0.5);
3543:C      5      info3 := trunc(ymin+0.5);
3544:C      5      info4 := trunc(ymax+0.5);
3545:C      5      gle_clip_limits (gle_gcb);
3546:C      5      end;
3547:S      3
3548:C      3      { recalculate the world to display transformation constants }
3549:S      3

```



```

3550:C      3      { x scale and offset
3551:C      3      with cur_disp_lim do
3552:C      4      temp := (xmax - xmin) / cur_vir_lim.xlim;
3553:S
3554:C      3      xwtod_scale := temp * ((viewport_lim.xmax - viewport_lim.xmin) /
3555:C      3      (window_lim.xmax - window_lim.xmin));
3556:S
3557:C      3      xwtod_offset := cur_disp_lim.xmin + (viewport_lim.xmin * temp) -
3558:C      3      (window_lim.xmin * xwtod_scale);
3559:S
3560:C      3      xdtow_scale := 1.0 / xwtod_scale;
3561:S
3562:C      3      xdtow_offset := - xwtod_offset;
3563:C      3      xwtod_offset := xwtod_offset + 0.5 {0.00000001};
3564:S
3565:C      3      { y scale and offset
3566:C      3      with cur_disp_lim do
3567:C      4      temp := (ymax - ymin) / cur_vir_lim.ylim;
3568:S
3569:C      3      ywtod_scale := temp * ((viewport_lim.ymax - viewport_lim.ymin) /
3570:C      3      (window_lim.ymax - window_lim.ymin));
3571:S
3572:C      3      ywtod_offset := cur_disp_lim.ymin + (viewport_lim.ymin * temp) -
3573:C      3      (window_lim.ymin * ywtod_scale);
3574:C      3      ydtow_scale := 1.0 / ywtod_scale;
3575:S
3576:C      3      ydtow_offset := - ywtod_offset;
3577:C      3      ywtod_offset := ywtod_offset + 0.5 {0.00000001};
3578:S
3579:C      3      if disp_eq_loc then
3580:C      4      begin
3581:C      5      {if display and locator are the same physical device then set
3582:C      5      the locator limits to the display limits
3583:C      5      }
3584:C      4      with cur_disp_lim do
3585:C      5      locator_limits(xmin,xmax,ymin,ymax);
3586:C      4      end
3587:C      4      else
3588:C      4      begin
3589:C      5      {set the locator limits to current locator limits
3590:C      5      }
3591:C      4      with log_loc_lim do
3592:C      5      locator_limits(xmin,xmax,ymin,ymax);
3593:C      4      end;
3594:C      3      { set up for int_move / int_line scale }
3595:C      3      int_test;
3596:C      3      end;
3597:C      2 end; { calculate_viewing }
3600:S
3601:S
3602:S
3603:D      -32 1 procedure display_limits ( dxmin, dxmax, dymin, dymax : real );
3604:S
3605:D      -32 2 { Purpose : To set the logical display limits in device units }
3606:S
3607:C      2 begin
3608:C      2 {WRITELN('NEW DISPLAY LIM = ',DXMIN:19:18,DXMAX:19:18,DYMIN:19:18,DYMAX:19:18);
3609:C      2 }

```

```

3610:C      2      with gcb^ do
3611:C      3      begin
3612:C      4      { set new log limits }
3613:C      4      with log_disp_lim do
3614:C      5      begin
3615:C      6      xmin := dxmin;
3616:C      6      xmax := dxmax;
3617:C      6      ymin := dymin;
3618:C      6      ymax := dymax;
3619:C      5      end;
3620:S
3621:C      3      { set the display width and height }
3622:S
3623:C      3      dxunits := dxmax - dxmin;
3624:C      3      dyunits := dymax - dymin;
3625:S
3626:C      3      { calculate aspect ratio of new log lim }
3627:C      3      log_aspect := dyunits / dxunits;
3628:S
3629:C      3      { set up new viewing transformation }
3630:C      3      calculate_viewing;
3631:C      3      end;
3632:C      2 end; { display_limits }
3633:S
3634:S
3635:C      1 end. { of module DGL_GEN }
3636:S
3637:C

```

No errors. No warnings.

***** Nonstandard language features enabled *****

GLE_FILE

Description

GLE_FILE contains file drivers which interface the graphics drivers to the file system.

Usage

Used with display spooling for HPGL device models. Spooling for non-HPGL devices is not supported in DGL.

Requirements

GLE_TYPES and GLE_UTLS.

Usage

Used in display spooling for HPGL device models. Spooling for non-HPGL devices is not supported in DGL.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:D      0 {
2:D      0 { Graphics Low End
3:D      0 {
4:D      0 { Module = GLE_FILE
5:D      0 { Programmer = BJS
6:D      0 { Date = 10-10-82
7:D      0 {
8:D      0 { Purpose: To provide general file drivers for ascii device handlers.
9:S      }
10:D     0 { Rev history
11:D     0 { Created = 10-10-82
12:D     0 { Modified = XX-XX-XX
13:S     }
14:S     {
15:S     { (c) Copyright Hewlett-Packard Company, 1983.
16:S     { All rights are reserved. Copying or other
17:S     { reproduction of this program except for archival
18:S     { purposes is prohibited without the prior
19:S     { written consent of Hewlett-Packard Company.
20:S     }
21:S     }
22:S     }
23:S     }
24:S     }
25:S     }
26:S     }
27:S     }
28:S     }
29:S     }
30:D     0 {
31:S     {
32:D     0 $SEARCH 'GLE_TYPES', 'GLE_UTILS'
33:D     0 $modcall$
34:D     0 $include 'OPTIONS' $ { ***** COMPILER OPTIONS ***** }
35:S     {
36:D     0 { This include file specifies range checking, debug and other compiler
37:S     { options for the graphics library
38:D     0 $debug OFF$
39:D     0 $range OFF$
40:D     0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'S
41:D     0 $FLOAT_HDW TEST$
42:S     }
43:S     }
44:S     }
45:S     }
46:D     0 $include 'OPTIONS' $ { ***** COMPILER OPTIONS ***** }
47:D     0 $LINENUM 9000$
9001:D   0 module gle_file_io;
9002:D   1 import gle_types;
9003:D   1 export
9004:D   1 type
9005:D   1 {
9006:D   1 { file_iocb_ptr = ^file_iocb;
9007:D   1 { file_iocb = record
9008:D   1 {
9009:D   1 { file_name : anyptr;
9010:D   1 {
9011:D   1 { name_size : gle_shortint;
9012:D   1 {

```

```

9013:D   1 { output_file : text;
9014:D   1 { lock_on_close : gle_shortint; { 1 = lock }
9015:D   1 { end;
9016:D   1 {
9017:D   1 { procedure file_init ( anyvar iocb_ptr : anyptr );
9018:D   1 { procedure file_write ( anyvar iocb_ptr, data_ptr : anyptr );
9019:D   1 { procedure file_inq_timeout ( anyvar iocb_ptr: anyptr; var value : integer );
9020:D   1 { procedure file_set_timeout ( anyvar iocb_ptr: anyptr; value : integer );
9021:D   1 { procedure file_term ( anyvar iocb_ptr : anyptr );
9022:D   1 {
9023:D   1 { implement
9024:D   1 {
9025:D   1 { import gle_utils;
9026:D   1 {
9027:D   1 { { The following types must match the types declared in GLE_HPGL, GLE_HPGLI }
9028:D   1 { type
9029:D   1 { {
9030:D   1 { { ascii_buffer_ptr = ^ascii_buffer;
9031:D   1 { {
9032:D   1 { { ascii_buffer = packed record
9033:D   1 { {
9034:D   1 { { maximum : integer;
9035:D   1 { { current : integer;
9036:D   1 { { data : packed array [1..32767] of char;
9037:D   1 { { end;
9038:D   1 {
9039:D   1 { procedure file_inq_timeout ( anyvar iocb_ptr : anyptr; var value : integer );
9040:D   2 { Dummy procedures for file io
9041:D   2 { begin
9042:D   2 { end;
9043:D   1 { procedure file_set_timeout ( anyvar iocb_ptr : anyptr; value : integer );
9044:D   2 { Dummy procedures for file io
9045:D   2 { begin
9046:D   2 { end;
9047:D   1 { procedure file_init ( anyvar iocb_ptr : anyptr );
9048:D   2 {
9049:D   2 { var
9050:D   2 { name : string[255];
9051:D   -256 2 {
9052:D   2 { begin
9053:D   2 { with file_iocb_ptr(iocb_ptr)^ do
9054:D   2 { begin
9055:D   3 { gle_copy_to_string(file_name, name_size, name);
9056:D   3 { rewrite(output_file, name);
9057:D   3 { end;
9058:D   2 { end;
9059:D   1 { procedure file_write ( anyvar iocb_ptr, data_ptr : anyptr );
9060:D   2 {
9061:D   2 { begin
9062:D   3 { with file_iocb_ptr(iocb_ptr)^, ascii_buffer_ptr(data_ptr)^ do
9063:D   3 { begin
9064:D   3 { writeln(output_file, data:current);
9065:D   3 { current := 0;
9066:D   3 { end;
9067:D   2 { end;
9068:D   1 { procedure file_term ( anyvar iocb_ptr : anyptr );
9069:D   2 {
9070:D   2 { begin
9071:D   2 { end;
9072:D   1 {

```



```
9073:S      2 begin
9074:C      2   with file_iocb_ptr(iocb_ptr)^ do
9075:C      3   begin
9076:C      3     if lock_on_close = 1 then close (output_file,'lock')
9077:C      4     else close (output_file);
9078:C      3   end;
9079:C      2 end;
9080:C      1 end. ( of module gle_file_io )
9081:S
9082:C
9083:S
```

No errors. No warnings.

***** Nonstandard language features enabled *****

GLE_GEN

Description

GLE_GEN provides an interface between the GLE caller and the output procedure variables in the GCB. It also contains a procedure used to initialize the GCB.

Requirements

GLE_TYPES.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:0      0 { Graphics Low End }
2:0      0 { }
3:0      0 { }
4:0      0 { Module = GLE_GEN }
5:0      0 { Programmer = BJS }
6:0      0 { Date = 10- 5-82 }
7:0      0 { }
8:0      0 { Purpose: To provide an interface between the GLE caller and the procedure }
9:0      0 { variables in the GCB. The module also initializes the GCB. }
10:0     0 { }
11:0     0 { Rev history }
12:0     0 { Created - 10- 5-82 }
13:0     0 { Modified - XX-XX-XX }
14:0     0 { }
15:0     0 { (c) Copyright Hewlett-Packard Company, 1983. }
16:0     0 { All rights are reserved. Copying or other }
17:0     0 { reproduction of this program except for archival }
18:0     0 { purposes is prohibited without the prior }
19:0     0 { written consent of Hewlett-Packard Company. }
20:0     0 { }
21:0     0 { }
22:0     0 { RESTRICTED RIGHTS LEGEND }
23:0     0 { }
24:0     0 { Use, duplication, or disclosure by the Government }
25:0     0 { is subject to restrictions as set forth in }
26:0     0 { paragraph (b) (3) (B) of the Rights in Technical }
27:0     0 { Data and Computer Software clause in }
28:0     0 { DAR 7-104.9(a). }
29:0     0 { }
30:0     0 { HEWLETT-PACKARD COMPANY }
31:0     0 { Fort Collins, Colorado }
32:0     0 { }
33:0     0 { }
34:0     0 { $modeals }
35:0     0 { $search 'GLE TYPES'S }
36:0     0 { $include 'OPTIONS'S ( ***** compiler options ***** ) }
37:0     0 { ( This include file specifies range checking, debug and other compiler }
38:0     0 { options for the graphics library }
39:0     0 { }
40:0     0 { $debug OFF$ }
41:0     0 { $range OFF$ }
42:0     0 { $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'S }
43:0     0 { $FLOAT_HDW TEST$ }
44:0     0 { }
45:0     0 { }
46:0     0 { }
47:0     0 { }
2000:0   0 { $include 'OPTIONS'S ( ***** compiler options ***** ) }
2001:0   0 { $linenum 2000$ }
2002:0   0 { }
2003:0   0 { module gle_gen; }
2004:0   1 { import gle_types; }
2005:0   1 { export }
2006:0   1 { }
2007:0   1 { }
2008:0   1 { procedure gle_move ( gcb : graphics_control_block_ptr ); }
2009:0   1 { procedure gle_draw ( gcb : graphics_control_block_ptr ); }
2010:0   1 { procedure gle_text ( gcb : graphics_control_block_ptr ); }
2011:0   1 { procedure gle_char_size ( gcb : graphics_control_block_ptr ); }

```

```

2012:0   1 { procedure gle_text_spacing ( gcb : graphics_control_block_ptr ); }
2013:0   1 { procedure gle_text_dir ( gcb : graphics_control_block_ptr ); }
2014:0   1 { procedure gle_text_just ( gcb : graphics_control_block_ptr ); }
2015:0   1 { procedure gle_marker ( gcb : graphics_control_block_ptr ); }
2016:0   1 { procedure gle_index_color ( gcb : graphics_control_block_ptr ); }
2017:0   1 { procedure gle_clear ( gcb : graphics_control_block_ptr ); }
2018:0   1 { procedure gle_clip_limits ( gcb : graphics_control_block_ptr ); }
2019:0   1 { procedure gle_init_gcb ( gcb : graphics_control_block_ptr ); }
2020:0   1 { procedure gle_get_pip2 ( gcb : graphics_control_block_ptr ); }
2021:0   1 { procedure gle_get_polygon_info ( gcb : graphics_control_block_ptr ); }
2022:0   1 { procedure gle_set_marker ( gcb : graphics_control_block_ptr ); }
2023:0   1 { procedure gle_marker_size ( gcb : graphics_control_block_ptr ); }
2024:0   1 { procedure gle_flush_buffer ( gcb : graphics_control_block_ptr ); }
2025:0   1 { procedure gle_term ( gcb : graphics_control_block_ptr ); }
2026:0   1 { procedure gle_cursor ( gcb : graphics_control_block_ptr ); }
2027:0   1 { procedure gle_polygon ( gcb : graphics_control_block_ptr ); }
2028:0   1 { procedure gle_fill_index_color ( gcb : graphics_control_block_ptr ); }
2029:0   1 { procedure gle_define_color_map ( gcb : graphics_control_block_ptr ); }
2030:0   1 { procedure gle_define_drawing_mode ( gcb : graphics_control_block_ptr ); }
2031:0   1 { procedure gle_linestyle ( gcb : graphics_control_block_ptr ); }
2032:0   1 { procedure gle_linewidth ( gcb : graphics_control_block_ptr ); }
2033:0   1 { procedure gle_buffer_mode ( gcb : graphics_control_block_ptr ); }
2034:0   1 { procedure gle_graphics_on_off ( gcb : graphics_control_block_ptr ); }
2035:0   1 { procedure gle_gload ( gcb : graphics_control_block_ptr ); }
2036:0   1 { procedure gle_gstore ( gcb : graphics_control_block_ptr ); }
2037:0   1 { procedure gle_get_raster ( gcb : graphics_control_block_ptr ); }
2038:0   1 { procedure gle_get_color_map ( gcb : graphics_control_block_ptr ); }
2039:0   1 { procedure gle_output_escapei ( gcb : graphics_control_block_ptr ); }
2040:0   1 { procedure gle_output_escapex ( gcb : graphics_control_block_ptr ); }
2041:0   1 { procedure gle_await_blanking ( gcb : graphics_control_block_ptr ); }
2042:0   1 { }
2043:0   1 { implement }
2044:0   1 { }
2045:0   1 { procedure gle_output_escapex ( gcb : graphics_control_block_ptr ); }
2046:0   2 { begin }
2047:0   2 { call (gcb^.output_escapex,gcb); }
2048:0   2 { end; }
2049:0   1 { }
2050:0   1 { procedure gle_output_escapex ( gcb : graphics_control_block_ptr ); }
2051:0   2 { begin }
2052:0   2 { call (gcb^.output_escapex,gcb); }
2053:0   2 { end; }
2054:0   1 { }
2055:0   1 { procedure gle_move ( gcb : graphics_control_block_ptr ); }
2056:0   2 { begin }
2057:0   2 { call (gcb^.move,gcb); }
2058:0   2 { end; }
2059:0   1 { }
2060:0   1 { procedure gle_draw ( gcb : graphics_control_block_ptr ); }
2061:0   2 { begin }
2062:0   2 { call (gcb^.draw,gcb); }
2063:0   2 { end; }
2064:0   1 { }
2065:0   1 { procedure gle_text ( gcb : graphics_control_block_ptr ); }
2066:0   2 { begin }
2067:0   2 { call (gcb^.draw,gcb); }
2068:0   2 { end; }
2069:0   1 { }
2070:0   1 { procedure gle_text ( gcb : graphics_control_block_ptr ); }
2071:0   2 { begin }

```



```

2072:C      2  call (gcb^.text,gcb);
2073:C      2  end;
2074:S
2075:D      1  procedure gle_char_size ( gcb : graphics_control_block_ptr );
2076:S
2077:C      2  begin
2078:C      2    call (gcb^.char_size,gcb);
2079:C      2  end;
2080:S
2081:D      1  procedure gle_text_spacing ( gcb : graphics_control_block_ptr );
2082:S
2083:C      2  begin
2084:C      2    call (gcb^.text_spacing,gcb);
2085:C      2  end;
2086:S
2087:D      1  procedure gle_text_dir ( gcb : graphics_control_block_ptr );
2088:S
2089:C      2  begin
2090:C      2    call (gcb^.text_dir,gcb);
2091:C      2  end;
2092:S
2093:D      1  procedure gle_clear ( gcb : graphics_control_block_ptr );
2094:S
2095:C      2  begin
2096:C      2    call (gcb^.clear,gcb);
2097:C      2  end;
2098:S
2099:D      1  procedure gle_text_just ( gcb : graphics_control_block_ptr );
2100:S
2101:C      2  begin
2102:C      2    call (gcb^.text_dir,gcb);
2103:C      2  end;
2104:S
2105:D      1  procedure gle_marker ( gcb : graphics_control_block_ptr );
2106:S
2107:C      2  begin
2108:C      2    call (gcb^.marker,gcb);
2109:C      2  end;
2110:S
2111:D      1  procedure gle_index_color ( gcb : graphics_control_block_ptr );
2112:S
2113:C      2  begin
2114:C      2    call (gcb^.index_color,gcb);
2115:C      2  end;
2116:S
2117:D      1  procedure gle_clip_limits ( gcb : graphics_control_block_ptr );
2118:S
2119:C      2  begin
2120:C      2    call (gcb^.clip_limits,gcb);
2121:C      2  end;
2122:S
2123:D      1  procedure gle_init_gcb ( gcb : graphics_control_block_ptr );
2124:S
2125:C      2  begin
2126:C      2    with gcb^ do
2127:C      3    begin
2128:C      3      display_name_char_count := 0;
2129:C      3      display_handler_char_count := 0;
2130:C      3      iocb := nil;
2131:C      3      device_buf := nil;

```

```

2132:C      3      dev_dep_stuff := nil;
2133:C      3      device_info := nil;
2134:C      3      error_return := 0;
2135:C      3      spooling := 0;
2136:C      3    end;
2137:C      2  end;
2138:S
2139:D      1  procedure gle_get_polygon_info ( gcb : graphics_control_block_ptr );
2140:S
2141:C      2  begin
2142:C      2    call(gcb^.get_polygon_info,gcb);
2143:C      2  end;
2144:S
2145:D      1  procedure gle_get_plp2 ( gcb : graphics_control_block_ptr );
2146:S
2147:C      2  begin
2148:C      2    call(gcb^.inq_plp2,gcb);
2149:C      2  end;
2150:S
2151:D      1  procedure gle_set_marker ( gcb : graphics_control_block_ptr );
2152:S
2153:C      2  begin
2154:C      2    call(gcb^.set_marker,gcb);
2155:C      2  end;
2156:S
2157:D      1  procedure gle_marker_size ( gcb : graphics_control_block_ptr );
2158:S
2159:C      2  begin
2160:C      2    call(gcb^.marker_size,gcb);
2161:C      2  end;
2162:S
2163:D      1  procedure gle_flush_buffer( gcb : graphics_control_block_ptr );
2164:S
2165:C      2  begin
2166:C      2    with gcb^ do
2167:C      3    call(gcb^.flush_buffer,gcb);
2168:C      2  end;
2169:S
2170:D      1  procedure gle_term ( gcb : graphics_control_block_ptr );
2171:S
2172:C      2  begin
2173:C      2    with gcb^ do
2174:C      3    call(io_term, iocb);
2175:C      2  end;
2176:S
2177:D      1  procedure gle_cursor ( gcb : graphics_control_block_ptr );
2178:S
2179:C      2  begin
2180:C      2    call (gcb^.cursor,gcb);
2181:C      2  end;
2182:S
2183:D      1  procedure gle_polygon ( gcb : graphics_control_block_ptr );
2184:S
2185:C      2  begin
2186:C      2    call (gcb^.polygon,gcb);
2187:C      2  end;
2188:S
2189:D      1  procedure gle_fill_index_color ( gcb : graphics_control_block_ptr );
2190:S
2191:C      2  begin

```



```

2192:C      2 call (gcb^.fill_index_color,gcb);
2193:C      2 end;
2194:S
2195:D      1 procedure gle_define_color_map( gcb : graphics_control_block_ptr );
2196:S
2197:C      2 begin
2198:C      2   call (gcb^.define_color_map,gcb);
2199:C      2 end;
2200:S
2201:D      1 procedure gle_define_drawing_mode ( gcb : graphics_control_block_ptr );
2202:S
2203:C      2 begin
2204:C      2   call (gcb^.define_drawing_mode,gcb);
2205:C      2 end;
2206:S
2207:D      1 procedure gle_linestyle ( gcb : graphics_control_block_ptr );
2208:S
2209:C      2 begin
2210:C      2   call (gcb^.linestyle,gcb);
2211:C      2 end;
2212:S
2213:D      1 procedure gle_linewidth ( gcb : graphics_control_block_ptr );
2214:S
2215:C      2 begin
2216:C      2   call (gcb^.linewidth,gcb);
2217:C      2 end;
2218:S
2219:D      1 procedure gle_buffer_mode ( gcb : graphics_control_block_ptr );
2220:S
2221:C      2 begin
2222:C      2   call (gcb^.buffer_mode,gcb);
2223:C      2 end;
2224:S
2225:D      1 procedure gle_graphics_on_off ( gcb : graphics_control_block_ptr );
2226:S
2227:C      2 begin
2228:C      2   call (gcb^.graphics_on_off,gcb);
2229:C      2 end;
2230:S
2231:D      1 procedure gle_gload ( gcb : graphics_control_block_ptr );
2232:S
2233:C      2 begin
2234:C      2   call (gcb^.gload,gcb);
2235:C      2 end;
2236:S
2237:D      1 procedure gle_gstore ( gcb : graphics_control_block_ptr );
2238:S
2239:C      2 begin
2240:C      2   call (gcb^.gstore,gcb);
2241:C      2 end;
2242:S
2243:D      1 procedure gle_get_raster ( gcb : graphics_control_block_ptr );
2244:S
2245:C      2 begin
2246:C      2   call (gcb^.get_raster,gcb);
2247:C      2 end;
2248:S
2249:D      1 procedure gle_get_color_map ( gcb : graphics_control_block_ptr );
2250:S
2251:C      2 begin

```

```

2252:C      2 call (gcb^.get_color_map,gcb);
2253:C      2 end;
2254:S
2255:D      1 procedure gle_await_blanking ( gcb : graphics_control_block_ptr );
2256:S
2257:C      2 begin
2258:C      2   call (gcb^.await_blanking,gcb);
2259:C      2 end;
2260:S
2261:S
2262:C      1 end. ( of module gle_gen )

```

No errors. No warnings.

***** Nonstandard language features enabled *****

GLE_GENI

Description

GLE_GENI provides an interface between the GLE caller and the input procedure variables in the GCB. It also contains a procedure to initialize the GCB.

Requirements

GLE_TYPES.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:D      0 {
2:D      0 { Graphics Low End }
3:D      0 {
4:D      0 { Module = GLE_GEN1 }
5:D      0 { Programmer = BJS }
6:D      0 { Date = 10-10-82 }
7:D      0 {
8:D      0 { Purpose: To provide interface procedure to the procedure variables in
9:D      0 { the GCB and to provide GCB initialization. }
10:S     0 {
11:D      0 { Rev history }
12:D      0 { Created - 10-10-82 }
13:D      0 { Modified - XX-XX-XX }
14:S     0 {
15:S     0 { (c) Copyright Hewlett-Packard Company, 1983.
16:S     0 { All rights are reserved. Copying or other
17:S     0 { reproduction of this program except for archival
18:S     0 { purposes is prohibited without the prior
19:S     0 { written consent of Hewlett-Packard Company.
20:S     0 {
21:S     0 {
22:S     0 {
23:S     0 {
24:S     0 {
25:S     0 {
26:S     0 {
27:S     0 {
28:S     0 {
29:S     0 {
30:S     0 {
31:D      0 {
32:S     0 {
33:D      0 $search 'GLE_TYPES'$
34:D      0 $modcall$
35:D      0 $include 'OPTIONS'$ { ***** COMPILER OPTIONS ***** }
36:S     0 { This include file specifies range checking, debug and other compiler
37:D      0 { options for the graphics library }
38:S     0 {
39:D      0 $debug OFF$
40:D      0 $range OFF$
41:D      0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
42:D      0 $FLOAT_HOW TEST$
43:S     0 {
44:S     0 {
45:S     0 {
46:S     0 {
47:D      0 $include 'OPTIONS'$ { ***** COMPILER OPTIONS ***** }
3000:D    0 $linenum 3000$
3001:S    0 {
3002:D    0 module gle_gen1;
3003:S    0 {
3004:D    1 import gle_types;
3005:S    0 {
3006:D    1 export
3007:S    0 {
3008:D    1 procedure gle_init_input_gcb ( gcbl : graphics_input_control_block_ptr );
3009:D    1 procedure gle_get_input_plp2 ( gcbl : graphics_input_control_block_ptr );
3010:D    1 procedure gle_input_term ( gcbl : graphics_input_control_block_ptr );
3011:D    1 procedure gle_input_echo ( gcbl : graphics_input_control_block_ptr );

```

```

3012:D    1 procedure gle_input_escape1 ( gcbl : graphics_input_control_block_ptr );
3013:D    1 procedure gle_input_escape0 ( gcbl : graphics_input_control_block_ptr );
3014:D    1 procedure gle_sample ( gcbl : graphics_input_control_block_ptr );
3015:D    1 procedure gle_start_digitize ( gcbl : graphics_input_control_block_ptr );
3016:D    1 procedure gle_get_digitize ( gcbl : graphics_input_control_block_ptr );
3017:S    0 {
3018:D    1 implement
3019:S    0 {
3020:D    1 procedure gle_init_input_gcb ( gcbl : graphics_input_control_block_ptr );
3021:S    0 {
3022:C    2 begin
3023:C    2 with gcbl^ do
3024:C    3 begin
3025:C    3 input_name_char_count := 0;
3026:C    3 input_handler_char_count := 0;
3027:S    0 {
3028:C    3 iocb := nil;
3029:C    3 device_buf := nil;
3030:C    3 dev_dev_stuff := nil;
3031:C    3 device_info := nil;
3032:C    3 error_return := 0;
3033:C    3 end;
3034:C    2 end;
3035:S    0 {
3036:D    1 procedure gle_get_input_plp2 ( gcbl : graphics_input_control_block_ptr );
3037:S    0 {
3038:C    2 begin
3039:C    2 call(gcbl^.inq_plp2,gcbl);
3040:C    2 end;
3041:S    0 {
3042:D    1 procedure gle_input_term ( gcbl : graphics_input_control_block_ptr );
3043:S    0 {
3044:C    2 begin
3045:C    2 with gcbl^ do
3046:C    3 call(io_term, iocb);
3047:C    2 end;
3048:S    0 {
3049:D    1 procedure gle_input_echo ( gcbl : graphics_input_control_block_ptr );
3050:S    0 {
3051:C    2 begin
3052:C    2 call (gcbl^.input_echo,gcbl);
3053:C    2 end;
3054:S    0 {
3055:D    1 procedure gle_input_escape1 ( gcbl : graphics_input_control_block_ptr );
3056:S    0 {
3057:C    2 begin
3058:C    2 call (gcbl^.input_escape1,gcbl);
3059:C    2 end;
3060:S    0 {
3061:D    1 procedure gle_input_escape0 ( gcbl : graphics_input_control_block_ptr );
3062:S    0 {
3063:C    2 begin
3064:C    2 call (gcbl^.input_escape0,gcbl);
3065:C    2 end;
3066:S    0 {
3067:D    1 procedure gle_sample ( gcbl : graphics_input_control_block_ptr );
3068:S    0 {
3069:C    2 begin
3070:C    2 call (gcbl^.sample,gcbl);
3071:C    2 end;

```



```
3072:S      1 procedure gle_start_digitize ( gcbi : graphics_input_control_block_ptr );
3073:D
3074:S
3075:C      2 begin
3076:C      2   call (gcbi^.start_digitize,gcbi);
3077:C      2 end;
3078:S
3079:D      1 procedure gle_get_digitize ( gcbi : graphics_input_control_block_ptr );
3080:S
3081:C      2 begin
3082:C      2   call (gcbi^.get_digitize,gcbi);
3083:C      2 end;
3084:S
3085:C      1 end. ( of module gle_geni )
```

No errors. No warnings.

***** Nonstandard language features enabled *****

GLE_HPGL

Description

GLE_HPGL contains the driver routines to drive HPGL output devices.

Requirements

GLE_TYPES, GLE_STEXT, GLE_ASTEXT, GLE_SCLIP, GLE_ASCLIP, GLE_SMARK, GLE_UTLS, and GLE_AUTL.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:D      0 { Graphics Low End }
2:D      0 {
3:D      0 { Module = GLE_HPGL_OUT
4:D      0 { Programmer = 8JS
5:D      0 { Date = 10- 5-82
6:D      0 {
7:D      0 { Purpose: To provide device dependent routines to drive hppl output
8:D      0 { plotters.
9:D      0 {
10:D     0 {
11:D     0 { Rev history
12:D     0 { Created - 10- 5-82 BJS
13:D     0 { Modified - 11-24-82 BJS removed refs to str routines
14:D     0 { 6-28-83 BJS added import of gle_astext (routine moved from
15:D     0 { gle_stext
16:D     0 { 3-14-84 80S added identifiers for 75868, 7550A, 7090, 7440
17:D     0 {
18:D     0 { (c) Copyright Hewlett-Packard Company, 1983.
19:D     0 { All rights are reserved. Copying or other
20:D     0 { reproduction of this program except for archival
21:D     0 { purposes is prohibited without the prior
22:D     0 { written consent of Hewlett-Packard Company.
23:D     0 {
24:D     0 {
25:D     0 {
26:D     0 {
27:D     0 {
28:D     0 {
29:D     0 {
30:D     0 {
31:D     0 {
32:D     0 {
33:D     0 {
34:D     0 {
35:D     0 {
36:D     0 { $search 'GLE_TYPES',
37:D     0 { 'GLE_SCLIP',
38:D     0 { 'ASH_SCLIP',
39:D     0 { 'GLE_STEXT',
40:D     0 { 'ASH_STEXT',
41:D     0 { 'GLE_SMARK',
42:D     0 { 'GLE_UTILS'$
43:D     0 { $modcal$
44:D     0 { $include 'OPTIONS'$ ( ***** COMPILER OPTIONS ***** )
45:D     0 { ( This include file specifies range checking, debug and other compiler
46:D     0 { options for the graphics library )
47:D     0 {
48:D     0 { $debug OFF$
49:D     0 { $range OFF$
50:D     0 { $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
51:D     0 { $FLCAT_HDW TEST$
52:D     0 {
53:D     0 {
54:D     0 {
55:D     0 {
56:D     0 { $include 'OPTIONS'$ ( ***** COMPILER OPTIONS ***** )
7000:D     0 { $irenum 7000$
7001:D     0 {
7002:D     0 { module GLE_HPGL_OUT;

```

```

7003:S      1 import gle_types;
7004:D      1
7005:S      1 export
7006:D      1
7007:S      1 const
7008:D      1 max_buffer = 255;
7009:D      1 buffer_fudge = 32;
7010:D      1
7011:S      1 type
7012:D      1 ascii_buffer_ptr = ^ascii_buffer;
7013:D      1
7014:S      1
7015:D      1 ascii_buffer = packed record
7016:D      1 maximum : integer;
7017:D      1 current : integer;
7018:D      1 data : packed array [1..max_buffer] of char;
7019:D      1 end;
7020:S      1
7021:D      1 driver_state_def = (moving,drawing,start_of_buffer,unknown);
7022:S      1
7023:D      1 hppl_device_rec_ptr = ^ hppl_device_rec;
7024:D      1 hppl_device_rec =
7025:D      1 record
7026:D      1 driver_state : driver_state_def;
7027:D      1 end;
7028:S      1
7029:D      1 procedure gle_init_hppl_output ( gcb : graphics_control_block_ptr );
7030:S      1
7031:D      1 implement
7032:S      1
7033:D      1 import gle_stext,
7034:D      1 gle_astext,
7035:D      1 gle_sclip,
7036:D      1 gle_smark,
7037:D      1 gle_utils;
7038:D      1
7039:S      1
7040:D      1 procedure hppl_flush_buffer ( gcb : graphics_control_block_ptr );
7041:S      1
7042:D      2 begin
7043:D      3 with gcb^,
7044:D      3 ascii_buffer_ptr(device_buf)^,
7045:D      3 hppl_device_rec_ptr(dev_dep_stuff)^ do
7046:D      3 begin
7047:D      3 if current <> 0 then call (io_write,iocb,device_buf);
7048:D      3 driver_state := start_of_buffer;
7049:D      3 end;
7050:D      2 end;
7051:S      1
7052:D      1 procedure dummy ( gcb : graphics_control_block_ptr );
7053:S      1
7054:D      2 begin
7055:D      2 end;
7056:S      1
7057:D      1 procedure buffer_cleanup ( gcb : graphics_control_block_ptr );
7058:S      1
7059:D      2 begin
7060:D      3 with gcb^,ascii_buffer_ptr(gcb^.device_buf)^ do
7061:D      3 if [current > maximum - buffer_fudge] or [current_buffer_mode = 0] then
7062:D      4 hppl_flush_buffer ( gcb );

```



```

7083:C      2 end;
7084:S
7085:D      1 procedure add_char_data ( gcb : graphics_control_block_ptr;
7086:D      2 count : gle_shortint;
7087:D      3 s : anychar_ptr );
7088:D      2 var
7089:D      -2 2 i : gle_shortint;
7090:S
7091:C      2 begin
7092:C      3 with gcb^,ascii_buffer_ptr(gcb^.device_buf)^ do
7093:C      4 begin
7094:C      5   for i := 1 to count do
7095:C      6     data[i+current] := s[i];
7096:C      7   current := current + count;
7097:C      8 end;
7098:C      2 end;
7099:S
7100:D      1 procedure add_parm_data ( gcb : graphics_control_block_ptr;
7101:D      2 value : gle_shortint);
7102:D      -2 2 count : gle_shortint;
7103:S
7104:C      2 begin
7105:C      3 with gcb^,ascii_buffer_ptr(gcb^.device_buf)^ do
7106:C      4 begin
7107:C      5   gle_write_integer (value,count,addr(data[current+1]));
7108:C      6   current := current + count;
7109:C      7 end;
7110:C      2 end;
7111:S
7112:D      1 procedure change_state ( gcb : graphics_control_block_ptr;
7113:D      2 new_state : driver_state_def);
7114:S
7115:D      -1 2 change : boolean;
7116:S
7117:C      2 begin
7118:C      3 with gcb^,hppl_device_rec_ptr(dev_dep_stuff)^ do
7119:C      4 begin
7120:C      5   change := new_state <> driver_state;
7121:C      6   if ((driver_state = unknown) or
7122:C      7     ((driver_state = moving) or (driver_state = drawing)) and change) then
7123:C      8     add_char_data(gcb,1,addr('; '));
7124:C      9   if change then
7125:C      10    begin
7126:C      11     driver_state := new_state;
7127:C      12     case driver_state of
7128:C      13     ***WARNING: (line 7110): 'ADDR' of a constant may not be supported on other implementations
7129:C      14     moving : add_char_data ( gcb, S, addr('PU;PA') );
7130:C      15     ***WARNING: (line 7111): 'ADDR' of a constant may not be supported on other implementations
7131:C      16     drawing : add_char_data ( gcb, S, addr('PD;PA') );
7132:C      17     start_of_buffer,unknown : ;
7133:C      18     end; { of case }
7134:C      19   end;
7135:C      20 end;
7136:C      2 end;
7137:S
7138:D      1 procedure hppl_output_escapeo ( gcb : graphics_control_block_ptr );
7139:D      2
7140:D      3
7141:D      4
7142:D      5
7143:D      6
7144:D      7
7145:D      8
7146:D      9
7147:D      10
7148:D      11
7149:D      12
7150:D      13
7151:D      14
7152:D      15
7153:D      16
7154:D      17
7155:D      18
7156:D      19
7157:D      20
7158:D      21
7159:D      22
7160:D      23
7161:D      24
7162:D      25
7163:D      26
7164:D      27
7165:D      28
7166:D      29
7167:D      30
7168:D      31
7169:D      32
7170:D      33
7171:D      34
7172:D      35
7173:D      36
7174:D      37
7175:D      38
7176:D      39
7177:D      40
7178:D      41
7179:D      42
7180:D      43
7181:D      44
7182:D      45
7183:D      46
7184:D      47
7185:D      48
7186:D      49
7187:D      50
7188:D      51
7189:D      52
7190:D      53
7191:D      54
7192:D      55
7193:D      56
7194:D      57
7195:D      58
7196:D      59
7197:D      60
7198:D      61
7199:D      62
7200:D      63
7201:D      64
7202:D      65
7203:D      66
7204:D      67
7205:D      68
7206:D      69
7207:D      70
7208:D      71
7209:D      72
7210:D      73
7211:D      74
7212:D      75
7213:D      76
7214:D      77
7215:D      78
7216:D      79
7217:D      80
7218:D      81
7219:D      82
7220:D      83
7221:D      84
7222:D      85
7223:D      86
7224:D      87
7225:D      88
7226:D      89
7227:D      90
7228:D      91
7229:D      92
7230:D      93
7231:D      94
7232:D      95
7233:D      96
7234:D      97
7235:D      98
7236:D      99
7237:D      100
7238:D      101
7239:D      102
7240:D      103
7241:D      104
7242:D      105
7243:D      106
7244:D      107
7245:D      108
7246:D      109
7247:D      110
7248:D      111
7249:D      112
7250:D      113
7251:D      114
7252:D      115
7253:D      116
7254:D      117
7255:D      118
7256:D      119
7257:D      120
7258:D      121
7259:D      122
7260:D      123
7261:D      124
7262:D      125
7263:D      126
7264:D      127
7265:D      128
7266:D      129
7267:D      130
7268:D      131
7269:D      132
7270:D      133
7271:D      134
7272:D      135
7273:D      136
7274:D      137
7275:D      138
7276:D      139
7277:D      140
7278:D      141
7279:D      142
7280:D      143
7281:D      144
7282:D      145
7283:D      146
7284:D      147
7285:D      148
7286:D      149
7287:D      150
7288:D      151
7289:D      152
7290:D      153
7291:D      154
7292:D      155
7293:D      156
7294:D      157
7295:D      158
7296:D      159
7297:D      160
7298:D      161
7299:D      162
7300:D      163
7301:D      164
7302:D      165
7303:D      166
7304:D      167
7305:D      168
7306:D      169
7307:D      170
7308:D      171
7309:D      172
7310:D      173
7311:D      174
7312:D      175
7313:D      176
7314:D      177
7315:D      178
7316:D      179
7317:D      180
7318:D      181
7319:D      182
7320:D      183
7321:D      184
7322:D      185
7323:D      186
7324:D      187
7325:D      188
7326:D      189
7327:D      190
7328:D      191
7329:D      192
7330:D      193
7331:D      194
7332:D      195
7333:D      196
7334:D      197
7335:D      198
7336:D      199
7337:D      200
7338:D      201
7339:D      202
7340:D      203
7341:D      204
7342:D      205
7343:D      206
7344:D      207
7345:D      208
7346:D      209
7347:D      210
7348:D      211
7349:D      212
7350:D      213
7351:D      214
7352:D      215
7353:D      216
7354:D      217
7355:D      218
7356:D      219
7357:D      220
7358:D      221
7359:D      222
7360:D      223
7361:D      224
7362:D      225
7363:D      226
7364:D      227
7365:D      228
7366:D      229
7367:D      230
7368:D      231
7369:D      232
7370:D      233
7371:D      234
7372:D      235
7373:D      236
7374:D      237
7375:D      238
7376:D      239
7377:D      240
7378:D      241
7379:D      242
7380:D      243
7381:D      244
7382:D      245
7383:D      246
7384:D      247
7385:D      248
7386:D      249
7387:D      250
7388:D      251
7389:D      252
7390:D      253
7391:D      254
7392:D      255
7393:D      256
7394:D      257
7395:D      258
7396:D      259
7397:D      260
7398:D      261
7399:D      262
7400:D      263
7401:D      264
7402:D      265
7403:D      266
7404:D      267
7405:D      268
7406:D      269
7407:D      270
7408:D      271
7409:D      272
7410:D      273
7411:D      274
7412:D      275
7413:D      276
7414:D      277
7415:D      278
7416:D      279
7417:D      280
7418:D      281
7419:D      282
7420:D      283
7421:D      284
7422:D      285
7423:D      286
7424:D      287
7425:D      288
7426:D      289
7427:D      290
7428:D      291
7429:D      292
7430:D      293
7431:D      294
7432:D      295
7433:D      296
7434:D      297
7435:D      298
7436:D      299
7437:D      300
7438:D      301
7439:D      302
7440:D      303
7441:D      304
7442:D      305
7443:D      306
7444:D      307
7445:D      308
7446:D      309
7447:D      310
7448:D      311
7449:D      312
7450:D      313
7451:D      314
7452:D      315
7453:D      316
7454:D      317
7455:D      318
7456:D      319
7457:D      320
7458:D      321
7459:D      322
7460:D      323
7461:D      324
7462:D      325
7463:D      326
7464:D      327
7465:D      328
7466:D      329
7467:D      330
7468:D      331
7469:D      332
7470:D      333
7471:D      334
7472:D      335
7473:D      336
7474:D      337
7475:D      338
7476:D      339
7477:D      340
7478:D      341
7479:D      342
7480:D      343
7481:D      344
7482:D      345
7483:D      346
7484:D      347
7485:D      348
7486:D      349
7487:D      350
7488:D      351
7489:D      352
7490:D      353
7491:D      354
7492:D      355
7493:D      356
7494:D      357
7495:D      358
7496:D      359
7497:D      360
7498:D      361
7499:D      362
7500:D      363
7501:D      364
7502:D      365
7503:D      366
7504:D      367
7505:D      368
7506:D      369
7507:D      370
7508:D      371
7509:D      372
7510:D      373
7511:D      374
7512:D      375
7513:D      376
7514:D      377
7515:D      378
7516:D      379
7517:D      380
7518:D      381
7519:D      382
7520:D      383
7521:D      384
7522:D      385
7523:D      386
7524:D      387
7525:D      388
7526:D      389
7527:D      390
7528:D      391
7529:D      392
7530:D      393
7531:D      394
7532:D      395
7533:D      396
7534:D      397
7535:D      398
7536:D      399
7537:D      400
7538:D      401
7539:D      402
7540:D      403
7541:D      404
7542:D      405
7543:D      406
7544:D      407
7545:D      408
7546:D      409
7547:D      410
7548:D      411
7549:D      412
7550:D      413
7551:D      414
7552:D      415
7553:D      416
7554:D      417
7555:D      418
7556:D      419
7557:D      420
7558:D      421
7559:D      422
7560:D      423
7561:D      424
7562:D      425
7563:D      426
7564:D      427
7565:D      428
7566:D      429
7567:D      430
7568:D      431
7569:D      432
7570:D      433
7571:D      434
7572:D      435
7573:D      436
7574:D      437
7575:D      438
7576:D      439
7577:D      440
7578:D      441
7579:D      442
7580:D      443
7581:D      444
7582:D      445
7583:D      446
7584:D      447
7585:D      448
7586:D      449
7587:D      450
7588:D      451
7589:D      452
7590:D      453
7591:D      454
7592:D      455
7593:D      456
7594:D      457
7595:D      458
7596:D      459
7597:D      460
7598:D      461
7599:D      462
7600:D      463
7601:D      464
7602:D      465
7603:D      466
7604:D      467
7605:D      468
7606:D      469
7607:D      470
7608:D      471
7609:D      472
7610:D      473
7611:D      474
7612:D      475
7613:D      476
7614:D      477
7615:D      478
7616:D      479
7617:D      480
7618:D      481
7619:D      482
7620:D      483
7621:D      484
7622:D      485
7623:D      486
7624:D      487
7625:D      488
7626:D      489
7627:D      490
7628:D      491
7629:D      492
7630:D      493
7631:D      494
7632:D      495
7633:D      496
7634:D      497
7635:D      498
7636:D      499
7637:D      500
7638:D      501
7639:D      502
7640:D      503
7641:D      504
7642:D      505
7643:D      506
7644:D      507
7645:D      508
7646:D      509
7647:D      510
7648:D      511
7649:D      512
7650:D      513
7651:D      514
7652:D      515
7653:D      516
7654:D      517
7655:D      518
7656:D      519
7657:D      520
7658:D      521
7659:D      522
7660:D      523
7661:D      524
7662:D      525
7663:D      526
7664:D      527
7665:D      528
7666:D      529
7667:D      530
7668:D      531
7669:D      532
7670:D      533
7671:D      534
7672:D      535
7673:D      536
7674:D      537
7675:D      538
7676:D      539
7677:D      540
7678:D      541
7679:D      542
7680:D      543
7681:D      544
7682:D      545
7683:D      546
7684:D      547
7685:D      548
7686:D      549
7687:D      550
7688:D      551
7689:D      552
7690:D      553
7691:D      554
7692:D      555
7693:D      556
7694:D      557
7695:D      558
7696:D      559
7697:D      560
7698:D      561
7699:D      562
7700:D      563
7701:D      564
7702:D      565
7703:D      566
7704:D      567
7705:D      568
7706:D      569
7707:D      570
7708:D      571
7709:D      572
7710:D      573
7711:D      574
7712:D      575
7713:D      576
7714:D      577
7715:D      578
7716:D      579
7717:D      580
7718:D      581
7719:D      582
7720:D      583
7721:D      584
7722:D      585
7723:D      586
7724:D      587
7725:D      588
7726:D      589
7727:D      590
7728:D      591
7729:D      592
7730:D      593
7731:D      594
7732:D      595
7733:D      596
7734:D      597
7735:D      598
7736:D      599
7737:D      600
7738:D      601
7739:D      602
7740:D      603
7741:D      604
7742:D      605
7743:D      606
7744:D      607
7745:D      608
7746:D      609
7747:D      610
7748:D      611
7749:D      612
7750:D      613
7751:D      614
7752:D      615
7753:D      616
7754:D      617
7755:D      618
7756:D      619
7757:D      620
7758:D      621
7759:D      622
7760:D      623
7761:D      624
7762:D      625
7763:D      626
7764:D      627
7765:D      628
7766:D      629
7767:D      630
7768:D      631
7769:D      632
7770:D      633
7771:D      634
7772:D      635
7773:D      636
7774:D      637
7775:D      638
7776:D      639
7777:D      640
7778:D      641
7779:D      642
7780:D      643
7781:D      644
7782:D      645
7783:D      646
7784:D      647
7785:D      648
7786:D      649
7787:D      650
7788:D      651
7789:D      652
7790:D      653
7791:D      654
7792:D      655
7793:D      656
7794:D      657
7795:D      658
7796:D      659
7797:D      660
7798:D      661
7799:D      662
7800:D      663
7801:D      664
7802:D      665
7803:D      666
7804:D      667
7805:D      668
7806:D      669
7807:D      670
7808:D      671
7809:D      672
7810:D      673
7811:D      674
7812:D      675
7813:D      676
7814:D      677
7815:D      678
7816:D      679
7817:D      680
7818:D      681
7819:D      682
7820:D      683
7821:D      684
7822:D      685
7823:D      686
7824:D      687
7825:D      688
7826:D      689
7827:D      690
7828:D      691
7829:D      692
7830:D      693
7831:D      694
7832:D      695
7833:D      696
7834:D      697
7835:D      698
7836:D      699
7837:D      700
7838:D      701
7839:D      702
7840:D      703
7841:D      704
7842:D      705
7843:D      706
7844:D      707
7845:D      708
7846:D      709
7847:D      710
7848:D      711
7849:D      712
7850:D      713
7851:D      714
7852:D      715
7853:D      716
7854:D      717
7855:D      718
7856:D      719
7857:D      720
7858:D      721
7859:D      722
7860:D      723
7861:D      724
7862:D      725
7863:D      726
7864:D      727
7865:D      728
7866:D      729
7867:D      730
7868:D      731
7869:D      732
7870:D      733
7871:D      734
7872:D      735
7873:D      736
7874:D      737
7875:D      738
7876:D      739
7877:D      740
7878:D      741
7879:D      742
7880:D      743
7881:D      744
7882:D      745
7883:D      746
7884:D      747
7885:D      748
7886:D      749
7887:D      750
7888:D      751
7889:D      752
7890:D      753
7891:D      754
7892:D      755
7893:D      756
7894:D      757
7895:D      758
7896:D      759
7897:D      760
7898:D      761
7899:D      762
7900:D      763
7901:D      764
7902:D      765
7903:D      766
7904:D      767
7905:D      768
7906:D      769
7907:D      770
7908:D      771
7909:D      772
7910:D      773
7911:D      774
7912:D      775
7913:D      776
7914:D      777
7915:D      778
7916:D      779
7917:D      780
7918:D      781
7919:D      782
7920:D      783
7921:D      784
7922:D      785
7923:D      786
7924:D      787
7925:D      788
7926:D      789
7927:D      790
7928:D      791
7929:D      792
7930:D      793
7931:D      794
7932:D      795
7933:D      796
7934:D      797
7935:D      798
7936:D      799
7937:D      800
7938:D      801
7939:D      802
7940:D      803
7941:D      804
7942:D      805
7943:D      806
7944:D      807
7945:D      808
7946:D      809
7947:D      810
7948:D      811
7949:D      812
7950:D      813
7951:D      814
7952:D      815
7953:D      816
7954:D      817
7955:D      818
7956:D      819
7957:D      820
7958:D      821
7959:D      822
7960:D      823
7961:D      824
7962:D      825
7963:D      826
7964:D      827
7965:D      828
7966:D      829
7967:D      830
7968:D      831
7969:D      832
7970:D      833
7971:D      834
7972:D      835
7973:D      836
7974:D      837
7975:D      838
7976:D      839
7977:D      840
7978:D      841
7979:D      842
7980:D      843
7981:D      844
7982:D      845
7983:D      846
7984:D      847
7985:D      848
7986:D      849
7987:D      850
7988:D      851
7989:D      852
7990:D      853
7991:D      854
7992:D      855
7993:D      856
7994:D      857
7995:D      858
7996:D      859
7997:D      860
7998:D      861
7999:D      862
8000:D      863
8001:D      864
8002:D      865
8003:D      866
8004:D      867
8005:D      868
8006:D      869
8007:D      870
8008:D      871
8009:D      872
8010:D      873
8011:D      874
8012:D      875
8013:D      876
8014:D      877
8015:D      878
8016:D      879
8017:D      880
8018:D      881
8019:D      882
8020:D      883
8021:D      884
8022:D      885
8023:D      886
8024:D      887
8025:D      888
8026:D      889
8027:D      890
8028:D      891
8029:D      892
8030:D      893
8031:D      894
8032:D      895
8033:D      896
8034:D      897
8035:D      898
8036:D      899
8037:D      900
8038:D      901
8039:D      902
8040:D      903
8041:D      904
8042:D      905
8043:D      906
8044:D      907
8045:D      908
8046:D      909
8047:D      910
8048:D      911
8049:D      912
8050:D      913
8051:D      914
8052:D      915
8053:D      916
8054:D      917
8055:D      918
8056:D      919
8057:D      920
8058:D      921
8059:D      922
8060:D      923
8061:D      924
8062:D      925
8063:D      926
8064:D      927
8065:D      928
8066:D      929
8067:D      930
8068:D      931
8069:D      932
8070:D      933
8071:D      934
8072:D      935
8073:D      936
8074:D      937
8075:D      938
8076:D      939
8077:D      940
8078:D      941
8079:D      942
8080:D      943
8081:D      944
8082:D      945
8083:D      946
8084:D      947
8085:D      948
8086:D      949
8087:D      950
8088:D      951
8089:D      952
8090:D      953
8091:D      954
8092:D      955
8093:D      956
8094:D      957
8095:D      958
8096:D      959
8097:D      960
8098:D      961
8099:D      962
8100:D      963
8101:D      964
8102:D      965
8103:D      966
8104:D      967
8105:D      968
8106:D      969
8107:D      970
8108:D      971
8109:D      972
8110:D      973
8111:D      974
8112:D      975
8113:D      976
8114:D      977
8115:D      978
8116:D      979
8117:D      980
8118:D      981
8119:D      982
8120:D      983
8121:D      984
8122:D      985
8123:D      986
8124:D      987
8125:D      988
8126:D      989
8127:D      990
8128:D      991
8129:D      992
8130:D      993
8131:D      994
8132:D      995
8133:D      996
8134:D      997
8135:D      998
8136:D      999
8137:D      1000

```

```

7120:C      2 begin
7121:C      3 with gcb^,hppl_device_rec_ptr(dev_dep_stuff)^ ,
7122:C      4   ascii_buffer_ptr(gcb^.device_buf)^ do
7123:C      5   begin
7124:C      6   try
7125:C      7   change_state ( gcb, unknown );
7126:C      8   add_char_data(gcb,info,anyptr(info_ptr1));
7127:C      9   buffer_cleanup ( gcb );
7128:C      10  recover
7129:C      11  if escapecode = -20 then
7130:C      12    begin
7131:C      13     current := 0;
7132:C      14     escape(-20);
7133:C      15    end;
7134:C      16  end;
7135:C      17 end;
7136:C      18 end;
713
```



```

7179:C      5      hpgl_flush_buffer ( gcb );
7180:C      5      call (io_read, iocb, device_buf);
7181:C      5      tcnt := 1;
7182:C      5      info1 := gle_read_integer (current,addr(data[1]),cnt);
7183:C      5      cnt := cnt + tcnt;
7184:C      5      info3 := gle_read_integer (current,addr(data[cnt]),tcnt);
7185:C      5      cnt := cnt + tcnt;
7186:C      5      info2 := gle_read_integer (current,addr(data[cnt]),tcnt);
7187:C      5      cnt := cnt + tcnt;
7188:C      5      info4 := gle_read_integer (current,addr(data[cnt]),tcnt);
7189:C      5      current := 0;
7190:C      5      if info1 > info2 then      { make xmin <= xmax }
7191:C      6      begin
7192:C      6      temp := info1;
7193:C      6      info1 := info2;
7194:C      6      info2 := temp;
7195:C      6      end;
7196:C      5      if info3 > info4 then      { make ymin <= ymax }
7197:C      6      begin
7198:C      6      temp := info3;
7199:C      6      info3 := info4;
7200:C      6      info4 := temp;
7201:C      6      end;
7202:C      5      end;
7203:C      5      else
7204:C      6      begin
7205:C      6      info1 := display_min_x;
7206:C      6      info2 := display_max_x;
7207:C      6      info3 := display_min_y;
7208:C      6      info4 := display_max_y;
7209:C      6      end;
7210:C      4      recover
7211:C      5      if escapecode = -20 then
7212:C      6      begin
7213:C      6      current := 0;
7214:C      6      escape(-20);
7215:C      6      end;
7216:C      5      end;
7217:C      2 end;
7218:C
7219:D
7220:S
7221:D
7222:D      2 var
7223:D      -4 2      tcnt,cnt      : gle_shortint;
7224:D      -12 2      tp1x,tp1y,tp2x,tp2y : gle_shortint;
7225:D
7226:S
7227:C      2 begin
7228:C      3      with gcb^,
7229:C      3      ascii_buffer_ptr(gcb^.device_buf)^,
7230:C      3      hpgl_device_rec_ptr(dev_dep_stuff)^ do
7231:C      3      begin
7232:C      3      try
***WARNING: (line 7232): 'ADDR' of a constant may not be supported on other implementations
7233:C      4      if (gle_match(4,addr(display_name),4,addr('7580')) or
***WARNING: (line 7233): 'ADDR' of a constant may not be supported on other implementations
7234:C      5      gle_match(4,addr(display_name),4,addr('7585')) or
***WARNING: (line 7234): 'ADDR' of a constant may not be supported on other implementations
7235:C      5      gle_match(4,addr(display_name),4,addr('7586')) then
7236:C      5      begin

```

```

7236:C      5      if spooling = 0 then
7237:C      6      begin
***WARNING: (line 7238): 'ADDR' of a constant may not be supported on other implementations
7238:C      6      add_char_data ( gcb, 2, addr('OH') );
7239:C      6      hpgl_flush_buffer ( gcb );
7240:C      6      call (io_read, iocb, device_buf);
7241:C      6      tcnt := 1;
7242:C      6      info1 := gle_read_integer (current,addr(data[1]),cnt);
7243:C      6      cnt := cnt + tcnt;
7244:C      6      info3 := gle_read_integer (current,addr(data[cnt]),tcnt);
7245:C      6      cnt := cnt + tcnt;
7246:C      6      info2 := gle_read_integer (current,addr(data[cnt]),tcnt);
7247:C      6      cnt := cnt + tcnt;
7248:C      6      info4 := gle_read_integer (current,addr(data[cnt]),tcnt);
7249:C      6      current := 0;
7250:C      6      end;
7251:C      6      else
7252:C      6      begin
***WARNING: (line 7253): 'ADDR' of a constant may not be supported on other implementations
7253:C      7      if (gle_match(4,addr(display_name),4,addr('7586')) or
***WARNING: (line 7254): 'ADDR' of a constant may not be supported on other implementations
7254:C      7      (gle_match(4,addr(display_name),4,addr('7585')) ) then
7255:C      7      begin
7256:C      7      info1 := -23656;      { E size }
7257:C      7      info2 := 23656;
7258:C      7      info3 := -17962;
7259:C      7      info4 := 17962;
7260:C      7      end;
7261:C      7      else
7262:C      7      begin
7263:C      7      info1 := -16190;      { D size }
7264:C      7      info2 := 16190;
7265:C      7      info3 := -10485;
7266:C      7      info4 := 10485;
7267:C      7      end;
7268:C      6      end;
7269:C      6      end;
7270:C      5      else
***WARNING: (line 7271): 'ADDR' of a constant may not be supported on other implementations
7271:C      5      if (gle_match(4,addr(display_name),4,addr('7470')) or
***WARNING: (line 7272): 'ADDR' of a constant may not be supported on other implementations
7272:C      6      (gle_match(4,addr(display_name),4,addr('7440')) ) then
7273:C      6      begin
7274:C      6      info1 := 0;
7275:C      6      info2 := 10300;
7276:C      6      info3 := 0;
7277:C      6      info4 := 7650;
7278:C      6      end;
7279:C      5      else
***WARNING: (line 7280): 'ADDR' of a constant may not be supported on other implementations
7280:C      6      if (gle_match(4,addr(display_name),4,addr('7475')) or
***WARNING: (line 7281): 'ADDR' of a constant may not be supported on other implementations
7281:C      7      (gle_match(4,addr(display_name),4,addr('7090')) ) then
7282:C      7      begin
7283:C      7      info1 := 0;
7284:C      7      info2 := 16640;
7285:C      7      info3 := 0;
7286:C      7      info4 := 10365;
7287:C      7      end;
7288:C      5      else

```



```
***WARNING: (line 7289): 'ADDR' of a constant may not be supported on other implementations
7289:C      7      if gle_match(4,addr(display_name),4,addr('7550')) then
7290:C      8      begin
7291:C      9      info1 := 0;
7292:C      8      info2 := 16450;
7293:C      8      info3 := 0;
7294:C      8      info4 := 10170;
7295:C      8      end
7296:C      9      else
***WARNING: (line 7297): 'ADDR' of a constant may not be supported on other implementations
7297:C      8      if gle_match(4,addr(display_name),4,addr('9872')) or (spooling = 1) then
7298:C      9      begin
7299:C      9      info1 := 0;
7300:C      9      info2 := 16000;
7301:C      9      info3 := 0;
7302:C      9      info4 := 11400;
7303:C      9      end
7304:C      9      else
7305:C      9      begin ( initialize the device and use P1/P2 values )
7306:C      9      hpgl_get_plp2 ( gcb );
7307:C      9      tp1x := info1; tp2x := info2;
7308:C      9      tp1y := info3; tp2y := info4;
***WARNING: (line 7309): 'ADDR' of a constant may not be supported on other implementations
7309:C      9      add_char_data ( gcb, 3, addr('IN')); ( BDS 3/28/84 )
7310:C      9      hpgl_flush_buffer ( gcb );
7311:C      9      hpgl_get_plp2 ( gcb );
7312:C      9      ( restore pl, p2 )
***WARNING: (line 7313): 'ADDR' of a constant may not be supported on other implementations
7313:C      9      add_char_data ( gcb, 2, addr('IP'));
7314:C      9      add_parm_data ( gcb, tp1x );
***WARNING: (line 7315): 'ADDR' of a constant may not be supported on other implementations
7315:C      9      add_char_data ( gcb, 1, addr(' '));
7316:C      9      add_parm_data ( gcb, tp1y );
***WARNING: (line 7317): 'ADDR' of a constant may not be supported on other implementations
7317:C      9      add_char_data ( gcb, 1, addr(' '));
7318:C      9      add_parm_data ( gcb, tp2x );
***WARNING: (line 7319): 'ADDR' of a constant may not be supported on other implementations
7319:C      9      add_char_data ( gcb, 1, addr(' '));
7320:C      9      add_parm_data ( gcb, tp2y );
7321:C      9      hpgl_flush_buffer ( gcb );
7322:C      9      end;
7323:C      4      recover
7324:C      4      if escapecode = -20 then
7325:C      5      begin
7326:C      5      current := 0;
7327:C      5      escape(-20);
7328:C      5      end;
7329:C      3      end;
7330:C      2      end;
7331:C      1      procedure hpgl_move ( gcb : graphics_control_block_ptr );
7332:C      2      begin
7333:C      2      with gcb^,hpgl_device_rec_ptr(dev_dep_stuff)^ ,
7334:C      3      ascii_buffer_ptr(device_buf)^ do
7335:C      3      begin
7336:C      3      try
7337:C      4      if driver_state = moving then
***WARNING: (line 7340): 'ADDR' of a constant may not be supported on other implementations
7340:C      5      add_char_data ( gcb, 1, addr(' '))
```

```
7341:C      5      else
7342:C      5      change_state ( gcb, moving );
7343:C      4      add_parm_data ( gcb, end_x );
***WARNING: (line 7344): 'ADDR' of a constant may not be supported on other implementations
7344:C      4      add_char_data ( gcb, 1, addr(' '));
7345:C      4      add_parm_data ( gcb, end_y );
7346:C      4      current_pos_x := end_x;
7347:C      4      current_pos_y := end_y;
7348:C      4      buffer_cleanup ( gcb );
7349:C      4      recover
7350:C      4      if escapecode = -20 then
7351:C      5      begin
7352:C      5      current := 0;
7353:C      5      escape(-20);
7354:C      5      end;
7355:C      3      end;
7356:C      2      end;
7357:C      1      procedure hpgl_draw ( gcb : graphics_control_block_ptr );
7358:C      2      begin
7359:C      2      with gcb^,hpgl_device_rec_ptr(dev_dep_stuff)^ ,
7360:C      3      ascii_buffer_ptr(device_buf)^ do
7361:C      3      begin
7362:C      3      try
7363:C      4      if driver_state = drawing then
***WARNING: (line 7366): 'ADDR' of a constant may not be supported on other implementations
7366:C      5      add_char_data ( gcb, 1, addr(' '));
7367:C      5      else
7368:C      5      change_state ( gcb, drawing );
7369:C      4      add_parm_data ( gcb, end_x );
***WARNING: (line 7370): 'ADDR' of a constant may not be supported on other implementations
7370:C      4      add_char_data ( gcb, 1, addr(' '));
7371:C      4      add_parm_data ( gcb, end_y );
7372:C      4      current_pos_x := end_x;
7373:C      4      current_pos_y := end_y;
7374:C      4      buffer_cleanup ( gcb );
7375:C      4      recover
7376:C      4      if escapecode = -20 then
7377:C      5      begin
7378:C      5      current := 0;
7379:C      5      escape(-20);
7380:C      5      end;
7381:C      3      end;
7382:C      2      end;
7383:C      1      procedure hpgl_buffer_mode( gcb : graphics_control_block_ptr );
7384:C      2      begin
7385:C      2      with gcb^,ascii_buffer_ptr(device_buf)^ do
7386:C      3      begin
7387:C      3      try
7388:C      4      hpgl_flush_buffer ( gcb );
7389:C      4      current_buffer_mode := info1;
7390:C      4      recover
7391:C      4      if escapecode = -20 then
7392:C      5      begin
7393:C      5      current := 0;
7394:C      5      escape(-20);
7395:C      5      end;
7396:C      5      end;
7397:C      5      end;
```



```

7398:C      3  end;
7399:C      2  end;
7400:S
7401:S      1  procedure hpgl_set_color ( gcb : graphics_control_block_ptr );
7402:S
7403:C      2  begin
7404:C      2  with gcb^,hpgl_device_rec_ptr(dev_dep_stuff)^ ,
7405:C      3  ascii_buffer_ptr(gcb^.device_buf)^ do
7406:C      3  begin
7407:C      3  try
7408:C      4  change_state ( gcb, unknown );
***WARNING: (line 7409): 'ADDR' of a constant may not be supported on other implementations
7409:C      4  add_char_data ( gcb, 5, addr('PU;5P') );
7410:C      4  add_parm_data ( gcb, info1 );
7411:C      4  buffer_cleanup ( gcb );
7412:C      4  current_color_index := info1;
7413:C      4  recover
7414:C      4  if escapecode = -20 then
7415:C      5  begin
7416:C      5  current := 0;
7417:C      5  escape(-20);
7418:C      5  end;
7419:C      3  end;
7420:C      2  end;
7421:S
7422:S      (***** this procedure is never used *****)
7423:S      procedure hpgl_fill_index_color ( gcb : graphics_control_block_ptr );
7424:S
7425:S      begin
7426:S      with gcb^,hpgl_device_rec_ptr(dev_dep_stuff)^ do
7427:S      current_polygon_color := info1;
7428:D      1  end;
7429:S
7430:S      1  procedure hpgl_linestyle ( gcb : graphics_control_block_ptr );
7431:S
7432:C      2  begin
7433:C      2  with gcb^,hpgl_device_rec_ptr(dev_dep_stuff)^ ,
7434:C      3  ascii_buffer_ptr(gcb^.device_buf)^ do
7435:C      3  begin
7436:C      3  try
7437:C      4  change_state ( gcb, unknown );
7438:C      4  current_linestyle := info1;
7439:C      4  current_pattern_length := info2;
7440:C      4  current_linestyle_mode := info3;
7441:C      4  current_linestyle_pattern := info4;
***WARNING: (line 7442): 'ADDR' of a constant may not be supported on other implementations
7442:C      4  add_char_data(gcb,2,addr('LT'));
7443:C      4  if info1 = 0 then
7444:C      5  begin
7445:C      5  end;
7446:C      5  else
7447:C      5  if info1 = 7 then
7448:C      6  add_parm_data(gcb,0)
7449:C      6  else
7450:C      6  begin
7451:C      6  if info3 = 0 then
7452:C      7  add_parm_data(gcb,info1)
7453:C      7  else
7454:C      7  add_parm_data(gcb,-info1);
***WARNING: (line 7455): 'ADDR' of a constant may not be supported on other implementations

```

```

7455:C      6  add_char_data(gcb,1,addr(' '));
7456:C      6  add_parm_data(gcb,info2);
7457:C      6  end;
7458:C      4  buffer_cleanup ( gcb );
7459:C      4  recover
7460:C      4  if escapecode = -20 then
7461:C      5  begin
7462:C      5  current := 0;
7463:C      5  escape(-20);
7464:C      5  end;
7465:C      3  end;
7466:C      2  end;
7467:S
7468:D      1  procedure hpgl_clear( gcb : graphics_control_block_ptr );
7469:S
7470:C      2  begin
7471:C      2  with gcb^,hpgl_device_rec_ptr(dev_dep_stuff)^ ,
7472:C      3  ascii_buffer_ptr(gcb^.device_buf)^ do
7473:C      3  begin
7474:C      3  try
7475:C      4  change_state ( gcb, unknown );
***WARNING: (line 7476): 'ADDR' of a constant may not be supported on other implementations
7476:C      4  add_char_data(gcb,2,addr('PG'));
7477:C      4  buffer_cleanup ( gcb );
7478:C      4  recover
7479:C      4  if escapecode = -20 then
7480:C      5  begin
7481:C      5  current := 0;
7482:C      5  escape(-20);
7483:C      5  end;
7484:C      3  end;
7485:C      2  end;
7486:S
7487:D      1  procedure hpgl_cursor ( gcb : graphics_control_block_ptr );
7488:S
7489:C      2  begin
7490:C      2  with gcb^,hpgl_device_rec_ptr(dev_dep_stuff)^ ,
7491:C      3  ascii_buffer_ptr(gcb^.device_buf)^ do
7492:C      3  begin
7493:C      3  try
7494:C      4  change_state ( gcb, unknown );
***WARNING: (line 7495): 'ADDR' of a constant may not be supported on other implementations
7495:C      4  add_char_data ( gcb, 5, addr('PU;PA') );
7496:C      4  add_parm_data ( gcb, info1 );
***WARNING: (line 7497): 'ADDR' of a constant may not be supported on other implementations
7497:C      4  add_char_data ( gcb, 1, addr(' '));
7498:C      4  add_parm_data ( gcb, info2);
7499:C      4  current_cursor_state := info3;
7500:C      4  hpgl_flush_buffer ( gcb );
7501:C      4  recover
7502:C      4  if escapecode = -20 then
7503:C      5  begin
7504:C      5  current := 0;
7505:C      5  escape(-20);
7506:C      5  end;
7507:C      3  end;
7508:C      2  end;
7509:S
7510:S      1  procedure gle_init_hpgl_output ( gcb : graphics_control_block_ptr);
7511:S

```



```

7512:D      2 var
7513:D      -4 2 saved_timeout : integer;
7514:D      -6 2 i : gle_shortint;
7515:S
7516:C      2 begin
7517:C      2 with gcb^,
7518:C      3 hpgl_device_rec_ptr(dev_dep_stuff)^,
7519:C      3 ascii_buffer_ptr(device_buf)^ do
7520:C      3 try
7521:C      4 current := 0;
7522:C      4 maximum := max_buffer;
7523:S
7524:C      4 driver_state := start_of_buffer;
7525:C      4 unclipped_move := hpgl_move;
7526:C      4 unclipped_draw := hpgl_draw;
7527:C      4 move := gle_soft_clip_move;
7528:C      4 draw := gle_soft_clip_draw;
7529:C      4 clear := hpgl_clear;
7530:C      4 text := gle_soft_text (hpgl_text);
7531:C      4 char_size := gle_soft_char_size;
7532:C      4 clip_limits := gle_soft_clip_limits;
7533:C      4 text_spacing := gle_soft_text_spacing;
7534:C      4 linestyle := hpgl_linestyle;
7535:C      4 text_dir := gle_soft_text_dir;
7536:C      4 text_just := gle_soft_text_just;
7537:C      4 marker := gle_soft_marker;
7538:C      4 marker_size := gle_soft_marker_size;
7539:C      4 set_marker := gle_soft_set_marker;
7540:C      4 index_color := hpgl_set_color;
7541:C      4 inq_pIp2 := hpgl_get_pIp2;
7542:C      4 get_polygon_info := dummy;
7543:C      4 graphics_on_off := dummy;
7544:C      4 cursor := hpgl_cursor;
7545:C      4 calc_soft_text_xform := gle_text_xform;
7546:C      4 buffer_mode := hpgl_buffer_mode;
7547:C      4 output_escapeo := hpgl_output_escapeo;
7548:C      4 output_escapei := hpgl_output_escapei;
7549:C      4 define_drawing_mode := dummy;
7550:C      4 define_color_map := dummy;
7551:C      4 polygon := dummy;
7552:C      4 fill_index_color := dummy;
7553:C      4 linewidth := dummy;
7554:C      4 gload := dummy;
7555:C      4 gstore := dummy;
7556:C      4 get_raster := dummy;
7557:C      4 get_color_map := dummy;
7558:C      4 await_blanking := dummy;
7559:C      4 flush_buffer := hpgl_flush_buffer;
7560:S
7561:C      4 soft_font_ptr := addr(font);
7562:S
7563:C      4 error_return := 0;
7564:C      4 if spooling = 0 then
7565:C      5 try
7566:C      6 call (io_inq_timeout, iocb, saved_timeout);
7567:C      6 call (io_set_timeout, iocb, 500 (ms));
7568:C      6 { send command that all HPGL plotters can respond to }
7569:C      6 { if the command fails then the address does not match }
7570:C      6 { the device }
***WARNING: (line 7571): 'ADDR' of a constant may not be supported on other implementations

```

```

7571:C      6 add_char_data ( gcb, 2, addr('OE') );
7572:C      6 hpgl_flush_buffer ( gcb );
7573:C      6 call (io_read, iocb, device_buf);
7574:C      6 current := 0;
7575:S
7576:C      6 { if this point is reached then a valid HPGL device was found }
7577:S
7578:C      6 try { perform an output identify seq. Note a 9872A will fail }
***WARNING: (line 7579): 'ADDR' of a constant may not be supported on other implementations
7579:C      7 add_char_data ( gcb, 2, addr('OI') );
7580:C      7 hpgl_flush_buffer ( gcb );
7581:C      7 call (io_read, iocb, device_buf);
7582:S
7583:C      7 for i := 1 to current do display_name[i] := data[i];
7584:C      7 for i := current+1 to 6 do display_name[i] := ' ';
7585:C      7 display_name_char_count := current;
7586:C      7 current := 0;
7587:C      7 recover
7588:C      7 if escapecode = -26 ( io system ) then
7589:C      8 begin
7590:C      9 display_name := '9872A ';
7591:C      9 display_name_char_count := 5;
7592:C      9 current := 0;
7593:C      8 end
7594:C      7 else if escapecode = -20 (stop key) then
7595:C      8 begin
7596:C      9 current := 0;
7597:C      9 escape(-20);
7598:C      8 end
7599:C      7 else escape(escapecode);
7600:S
7601:C      7 recover
7602:C      7 if escapecode = -26 then error_return := 1
7603:C      7 else if escapecode = -20 (stop key) then
7604:C      8 begin
7605:C      9 current := 0;
7606:C      9 escape(-20);
7607:C      8 end
7608:C      7 else escape(escapecode); { ignore io errors }
7609:S
7610:C      4 call (io_set_timeout, iocb, saved_timeout);
7611:S
7612:C      4 if error_return = 0 then
7613:C      5 begin
***WARNING: (line 7614): 'ADDR' of a constant may not be supported on other implementations
7614:C      5 add_char_data ( gcb, 12, addr('DF;SPI;IM30;') );
7615:C      5 hpgl_flush_buffer ( gcb );
***WARNING: (line 7616): 'ADDR' of a constant may not be supported on other implementations
7616:C      5 if gle_match(4,addr(display_name),4,addr('9111')) then { 9111 is input only }
7617:C      6 escape(-26) { force io error }
7618:C      5 else
***WARNING: (line 7619): 'ADDR' of a constant may not be supported on other implementations
7619:C      6 if gle_match(display_name_char_count,addr(display_name),5,addr('7470A')) or
***WARNING: (line 7620): 'ADDR' of a constant may not be supported on other implementations
7620:C      6 gle_match(display_name_char_count,addr(display_name),4,addr('7470')) then
7621:C      7 begin
7622:C      8 palette := 2;
7623:C      8 cont_linestyles := 8;
7624:C      8 vect_linestyles := 0;
7625:C      7 end

```



```

7626:C      7      else
***WARNING: (line 7627): 'ADDR' of a constant may not be supported on other implementations
7627:C      7      if gle_match(display_name_char_count,addr(display_name),4,addr('7440')) then
7628:C      8      begin
7629:C      8      palette := 8;
7630:C      8      cont_linestyles := 8;
7631:C      8      vect_linestyles := 0;
7632:C      8      end
7633:C      8      else
***WARNING: (line 7634): 'ADDR' of a constant may not be supported on other implementations
7634:C      8      if gle_match(display_name_char_count,addr(display_name),5,addr('7475A')) or
***WARNING: (line 7635): 'ADDR' of a constant may not be supported on other implementations
7635:C      9      gle_match(display_name_char_count,addr(display_name),4,addr('7475')) or
***WARNING: (line 7636): 'ADDR' of a constant may not be supported on other implementations
7636:C      9      gle_match(display_name_char_count,addr(display_name),4,addr('7090')) then
7637:C      9      begin
7638:C      9      palette := 6;
7639:C      9      cont_linestyles := 8;
7640:C      9      vect_linestyles := 0;
7641:C      9      end
7642:C      9      else
***WARNING: (line 7643): 'ADDR' of a constant may not be supported on other implementations
7643:C      9      if (gle_match(display_name_char_count,addr(display_name),5,addr('7580A'))) or
***WARNING: (line 7644): 'ADDR' of a constant may not be supported on other implementations
7644:C      10     (gle_match(display_name_char_count,addr(display_name),5,addr('7580B'))) or
***WARNING: (line 7645): 'ADDR' of a constant may not be supported on other implementations
7645:C      10     (gle_match(display_name_char_count,addr(display_name),4,addr('7580'))) or
***WARNING: (line 7646): 'ADDR' of a constant may not be supported on other implementations
7646:C      10     (gle_match(display_name_char_count,addr(display_name),5,addr('7585A'))) or
***WARNING: (line 7647): 'ADDR' of a constant may not be supported on other implementations
7647:C      10     (gle_match(display_name_char_count,addr(display_name),5,addr('7585B'))) or
***WARNING: (line 7648): 'ADDR' of a constant may not be supported on other implementations
7648:C      10     (gle_match(display_name_char_count,addr(display_name),4,addr('7585'))) or
***WARNING: (line 7649): 'ADDR' of a constant may not be supported on other implementations
7649:C      10     (gle_match(display_name_char_count,addr(display_name),5,addr('7586B'))) or
***WARNING: (line 7650): 'ADDR' of a constant may not be supported on other implementations
7650:C      10     (gle_match(display_name_char_count,addr(display_name),5,addr('7586'))) or
***WARNING: (line 7651): 'ADDR' of a constant may not be supported on other implementations
7651:C      10     (gle_match(display_name_char_count,addr(display_name),5,addr('7550A'))) or
***WARNING: (line 7652): 'ADDR' of a constant may not be supported on other implementations
7652:C      10     (gle_match(display_name_char_count,addr(display_name),5,addr('7550'))) then
7653:C      10     begin
7654:C      10     palette := 8;
7655:C      10     cont_linestyles := 8;
7656:C      10     vect_linestyles := 8;
7657:C      10     end
7658:C      10     else
7659:C      10     begin
***WARNING: (line 7660): 'ADDR' of a constant may not be supported on other implementations
7660:C      10     if (gle_match(display_name_char_count,addr(display_name),5,addr('9872A'))) or
***WARNING: (line 7661): 'ADDR' of a constant may not be supported on other implementations
7661:C      11     (gle_match(display_name_char_count,addr(display_name),5,addr('9872B'))) or
***WARNING: (line 7662): 'ADDR' of a constant may not be supported on other implementations
7662:C      11     (gle_match(display_name_char_count,addr(display_name),5,addr('98725'))) or
***WARNING: (line 7663): 'ADDR' of a constant may not be supported on other implementations
7663:C      11     (gle_match(display_name_char_count,addr(display_name),4,addr('9872'))) then
7664:C      11     palette := 4;
7665:C      11     else
***WARNING: (line 7666): 'ADDR' of a constant may not be supported on other implementations
7666:C      11     if (gle_match(display_name_char_count,addr(display_name),5,addr('9872C'))) or

```

```

***WARNING: (line 7667): 'ADDR' of a constant may not be supported on other implementations
7667:C      12     (gle_match(display_name_char_count,addr(display_name),5,addr('9872T'))) then
7668:C      12     palette := 8;
7669:C      12     else
7670:C      12     if spooling = 0 then palette := 4 { assume 9872 like device }
7671:C      13     else
7672:C      13     escape(-26); { can't init device, force io error }
7673:C      13
7674:C      10     cont_linestyles := 8;
7675:C      10     vect_linestyles := 0;
7676:C      10     end;
7677:C      5
7678:C      5     hpgl_get_hard_clip ( gcb );
7679:C      5     display_min_x := info1;
7680:C      5     display_min_y := info3;
7681:C      5     display_max_x := info2;
7682:C      5     display_max_y := info4;
7683:C      5
7684:C      5     gle_soft_clip_limits ( gcb ); { set default clipping limits }
7685:C      5
7686:C      5     gamut := palette;
7687:C      5     polygon_support := 0; { polygon routine dummyed out }
7688:C      5     display_handler_name := 'HPGL';
7689:C      5     display_handler_char_count := 4;
7690:C      5
7691:C      5     display_res_x := 40;
7692:C      5     display_res_y := 40;
7693:C      5
7694:C      5     linewidths := 1;
7695:C      5     char_sizes := -1;
7696:C      5     background := 0;
7697:C      5     complement_support := 0;
7698:C      5     non_dominant_support := 0;
7699:C      5     erase_support := 0;
7700:C      5     color_map_support := 0;
7701:C      5     redef_background := 0;
7702:C      5
7703:C      5     polygon_fill_factor := 16;
7704:C      5     polygon_solid_fill := 5;
7705:C      5     dither_support := 0;
7706:C      5
7707:C      5     current_pos_x := 0;
7708:C      5     current_pos_y := 0;
7709:C      5     current_cursor_state := 0; { off }
7710:C      5
7711:C      5     current_buffer_mode := 0; { imed visb }
7712:C      5     end;
7713:C      4     recover
7714:C      4     begin
7715:C      4     if escapecode = -26 then error_return := 1
7716:C      5     else if escapecode = -20 {stop key} then
7717:C      6     begin
7718:C      6     current := 0;
7719:C      6     escape(-20);
7720:C      6     end
7721:C      6     else escape(escapecode); { ignor io errors }
7722:C      4     end;
7723:C      2 end;
7724:C      5
7725:C      1 end. { hpgl_output }

```


7726:S

No errors. 57 warnings.

***** Nonstandard language features enabled *****

GLE_HPGLI

Description

GLE_HPGLI contains the driver code for HPGL input devices.

Requirements

GLE_TYPES, and GLE_UTLS.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:D      0 {
2:D      0 { Graphics Low End
3:D      0 {
4:D      0 { Module = GLE_HPGLI.IN
5:D      0 { Programmer = BJS
6:D      0 { Date = 10-10-82
7:D      0 {
8:D      0 { Purpose: To provide HPGL input handler routines.
9:S      }
10:D     0 { Rev history
11:D     0 { Created = 10-10-82
12:D     0 { Modified = 03-15-84 BDS Added support for 7586B, 7550A, 7440, 7090
13:S     }
14:S     {
15:S     { (c) Copyright Hewlett-Packard Company, 1983.
16:S     { All rights are reserved. Copying or other
17:S     { reproduction of this program except for archival
18:S     { purposes is prohibited without the prior
19:S     { written consent of Hewlett-Packard Company.
20:S     }
21:S     }
22:S     }
23:S     }
24:S     }
25:S     }
26:S     }
27:S     }
28:S     }
29:S     }
30:D     0 {
31:S     }
32:S     }
33:D     0 $SEARCH 'GLE_TYPES','GLE_UTILS','GEN_TOOLS'$
34:D     0 $modcal$
35:D     0 $INCLUDE 'OPTIONS'$ { ***** COMPILER OPTIONS ***** }
36:S     {
37:D     0 { This include file specifies range checking, debug and other compiler
38:S     { options for the graphics library
39:D     }
40:D     0 $debug OFF$
41:D     0 $range OFF$
42:D     0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
43:D     0 $FLOAT_HDW TEST$
44:S     }
45:S     }
46:S     }
47:D     0 $INCLUDE 'OPTIONS'$ { ***** COMPILER OPTIONS ***** }
18000:D  0 $LINENUM 18000$
18001:S  0
18002:D  0 module gle_hppl_in;
18003:S  0
18004:D  1 import gle_types;
18005:S  1
18006:D  1 export
18007:S  1
18008:D  1 const
18009:D  1   max_buffer = 255;
18010:D  1   buffer_fudge = 32;
18011:S  1

```

```

18012:D  1 type
18013:D  1   ascii_buffer_ptr = ^ascii_buffer;
18014:S  1
18015:D  1   ascii_buffer = packed record
18016:D  1     maximum : integer;
18017:D  1     current : integer;
18018:D  1     data : packed array [1..max_buffer] of char;
18019:D  1   end;
18020:S  1
18021:D  1 procedure gle_init_hppl_input ( gcbi : graphics_input_control_block_ptr);
18022:S  1
18023:D  1 implement
18024:S  1
18025:D  1   import gle_utils(
18026:D  1     gen_tools);
18027:S  1
18028:D  1 procedure hppl_flush_buffer ( gcbi : graphics_input_control_block_ptr );
18029:S  1
18030:D  2 begin
18031:D  2   with gcbi^,
18032:D  2     ascii_buffer_ptr(device_buf)^ do
18033:D  2     begin
18034:D  2       if current <> 0 then call (io_write,iocb,device_buf);
18035:D  2     end;
18036:D  2 end;
18037:S  2
18038:D  1 procedure add_char_data ( gcbi : graphics_input_control_block_ptr;
18039:D  2   count : gle_shortint;
18040:D  2   s : anychar_ptr );
18041:D  2 var
18042:D  2   i : gle_shortint;
18043:S  2
18044:D  2 begin
18045:D  2   with gcbi^,ascii_buffer_ptr(gcbi^.device_buf)^ do
18046:D  2   begin
18047:D  2     for i := 1 to count do
18048:D  2       data[i+current] := s[i];
18049:D  2     current := current + count;
18050:D  2   end;
18051:D  2 end;
18052:S  2
18053:D  1 procedure add_parm_data ( gcbi : graphics_input_control_block_ptr;
18054:D  2   value : gle_shortint);
18055:D  2 var
18056:D  2   count : gle_shortint;
18057:S  2
18058:D  2 begin
18059:D  2   with gcbi^,ascii_buffer_ptr(gcbi^.device_buf)^ do
18060:D  2   begin
18061:D  2     gle_write_integer (value,count,addr(data[current+1]));
18062:D  2     current := current + count;
18063:D  2   end;
18064:D  2 end;
18065:S  2
18066:D  1 procedure hppl_input_escapeo ( gcbi : graphics_input_control_block_ptr );
18067:S  1
18068:D  2 begin
18069:D  2   with gcbi^ do
18070:D  2   begin
18071:D  2     add_char_data(gcbi,info1,anyptr(info_ptr1));

```



```

18072:C      3      hpgl_flush_buffer ( gcbl );
18073:C      3      end;
18074:C      2      end;
18075:S
18076:D      1      procedure hpgl_input_escape1 ( gcbl : graphics_input_control_block_ptr );
18077:S
18078:D      2      var
18079:D      -2      i : gle_shortint;
18080:D      -6      sptr : anychar_ptr;
18081:S
18082:C      2      begin
18083:C      2      with gcbl^,ascii_buffer_ptr(gcbl^.device_buf)^ do
18084:C      2      begin
18085:C      2      call (io_read,iocb, device_buf);
18086:C      2      sptr := anyptr(info_ptr1);
18087:C      2      info1 := current;
18088:C      2      for i := 1 to current do sptr[i] := data[i];
18089:C      2      current := 0;
18090:C      2      end;
18091:C      2      end;
18092:S
18093:D      1      procedure hpgl_get_input_plp2 ( gcbl : graphics_input_control_block_ptr );
18094:S
18095:D      2      var
18096:D      -2      cnt : gle_shortint;
18097:D      -4      tcnt : gle_shortint;
18098:D      -8      temp : integer;
18099:S
18100:C      2      begin
18101:C      2      with gcbl^,
18102:C      2      ascii_buffer_ptr(gcbl^.device_buf)^ do
18103:C      2      begin
***WARNING: (line18104): 'ADDR' of a constant may not be supported on other implementations
18104:C      3      add_char_data ( gcbl, 2, addr('OP') );
18105:C      3      hpgl_flush_buffer ( gcbl );
18106:C      3      call (io_read,iocb, device_buf);
18107:C      3      tcnt := 1;
18108:C      3      info1 := gle_read_integer (current,addr(data[1]),cnt);
18109:C      3      cnt := cnt + tcnt;
18110:C      3      info3 := gle_read_integer (current,addr(data[cnt]),tcnt);
18111:C      3      cnt := cnt + tcnt;
18112:C      3      info2 := gle_read_integer (current,addr(data[cnt]),tcnt);
18113:C      3      cnt := cnt + tcnt;
18114:C      3      info4 := gle_read_integer (current,addr(data[cnt]),tcnt);
18115:C      3      current := 0;
18116:C      3      if info1 > info2 then      { make xmin <= xmax }
18117:C      4      begin
18118:C      4      temp := info1;
18119:C      4      info1 := info2;
18120:C      4      info2 := temp;
18121:C      4      end;
18122:C      3      if info3 > info4 then      { make ymin <= ymax }
18123:C      4      begin
18124:C      4      temp := info3;
18125:C      4      info3 := info4;
18126:C      4      info4 := temp;
18127:C      4      end;
18128:C      3      end;
18129:C      2      end;
18130:S

```

```

18131:D      1      procedure hpgl_get_input_hard_clip ( gcb : graphics_input_control_block_ptr );
18132:S
18133:D      2      var
18134:D      -4      tcnt,cnt : gle_shortint;
18135:D      -12      tpx,tply,tp2x,tp2y : gle_shortint;
18136:S
18137:C      2      begin
18138:C      2      with gcb^,
18139:C      3      ascii_buffer_ptr(gcb^.device_buf)^ do
***WARNING: (line18140): 'ADDR' of a constant may not be supported on other implementations
18140:C      3      if gle_match(4,addr(input_name),4,addr('7580')) or
***WARNING: (line18141): 'ADDR' of a constant may not be supported on other implementations
18141:C      3      gle_match(4,addr(input_name),4,addr('7585')) or
***WARNING: (line18142): 'ADDR' of a constant may not be supported on other implementations
18142:C      4      gle_match(4,addr(input_name),4,addr('7586')) then
18143:C      4      begin
***WARNING: (line18144): 'ADDR' of a constant may not be supported on other implementations
18144:C      4      add_char_data ( gcb, 2, addr('OH') );
18145:C      4      hpgl_flush_buffer ( gcb );
18146:C      4      call (io_read,iocb, device_buf);
18147:C      4      tcnt := 1;
18148:C      4      info1 := gle_read_integer (current,addr(data[1]),cnt);
18149:C      4      cnt := cnt + tcnt;
18150:C      4      info3 := gle_read_integer (current,addr(data[cnt]),tcnt);
18151:C      4      cnt := cnt + tcnt;
18152:C      4      info2 := gle_read_integer (current,addr(data[cnt]),tcnt);
18153:C      4      cnt := cnt + tcnt;
18154:C      4      info4 := gle_read_integer (current,addr(data[cnt]),tcnt);
18155:C      4      current := 0;
18156:C      4      end
18157:C      4      else
***WARNING: (line18158): 'ADDR' of a constant may not be supported on other implementations
18158:C      4      if gle_match(4,addr(input_name),4,addr('7470')) or
***WARNING: (line18159): 'ADDR' of a constant may not be supported on other implementations
18159:C      4      gle_match(4,addr(input_name),4,addr('7440')) then
18160:C      5      begin
18161:C      5      info1 := 0;
18162:C      5      info2 := 10300;
18163:C      5      info3 := 0;
18164:C      5      info4 := 7650;
18165:C      5      end
18166:C      4      else
***WARNING: (line18167): 'ADDR' of a constant may not be supported on other implementations
18167:C      5      if gle_match(4,addr(input_name),4,addr('7475')) or
***WARNING: (line18168): 'ADDR' of a constant may not be supported on other implementations
18168:C      6      gle_match(4,addr(input_name),4,addr('7090')) then
18169:C      6      begin
18170:C      6      info1 := 0;
18171:C      6      info2 := 16640;
18172:C      6      info3 := 0;
18173:C      6      info4 := 10365;
18174:C      6      end
18175:C      4      else
***WARNING: (line18176): 'ADDR' of a constant may not be supported on other implementations
18176:C      5      if gle_match(4,addr(input_name),4,addr('7550')) then
18177:C      6      begin
18178:C      6      info1 := 0;
18179:C      6      info2 := 16450;
18180:C      6      info3 := 0;
18181:C      6      info4 := 10170;

```



```

18182:C      7      end
18183:C      7      else
***WARNING: (line18184): 'ADDR' of a constant may not be supported on other implementations
18184:C      7      if gle_match(4,addr(input_name),4,addr('9872')) then
18185:C      8      begin
18186:C      8          info1 := 0;
18187:C      8          info2 := 16000;
18188:C      8          info3 := 0;
18189:C      8          info4 := 11400;
18190:C      8      end
18191:C      8      else
***WARNING: (line18192): 'ADDR' of a constant may not be supported on other implementations
18192:C      8      if gle_match(4,addr(input_name),4,addr('9111')) then
18193:C      9      begin
18194:C      9          info1 := 0;
18195:C      9          info2 := 12032;
18196:C      9          info3 := 0;
18197:C      9          info4 := 8704;
18198:C      9      end
18199:C      9      else
18200:C      9      begin ( initialize the device and use P1/P2 values )
18201:C      9          hpgl_get_input_plp2 ( gcbl );
18202:C      9          tp1x := info1; tp2x := info2;
18203:C      9          tp1y := info3; tp2y := info4;
***WARNING: (line18204): 'ADDR' of a constant may not be supported on other implementations
18204:C      9          add_char_data ( gcbl, 2, addr('IN'));
18205:C      9          hpgl_get_input_plp2 ( gcbl );
18206:C      9          ( restore p1, p2 )
***WARNING: (line18207): 'ADDR' of a constant may not be supported on other implementations
18207:C      9          add_char_data ( gcbl, 2, addr('IP'));
18208:C      9          add_parm_data ( gcbl, tp1x );
***WARNING: (line18209): 'ADDR' of a constant may not be supported on other implementations
18209:C      9          add_char_data ( gcbl, 1, addr(' '));
18210:C      9          add_parm_data ( gcbl, tp1y );
***WARNING: (line18211): 'ADDR' of a constant may not be supported on other implementations
18211:C      9          add_char_data ( gcbl, 1, addr(' '));
18212:C      9          add_parm_data ( gcbl, tp2x );
***WARNING: (line18213): 'ADDR' of a constant may not be supported on other implementations
18213:C      9          add_char_data ( gcbl, 1, addr(' '));
18214:C      9          add_parm_data ( gcbl, tp2y );
18215:C      9          hpgl_flush_buffer ( gcbl );
18216:C      9      end;
18217:C      2 end;
18218:S
18219:S
18220:D      1 procedure hpgl_sample ( gcbl : graphics_input_control_block_ptr );
18221:S
18222:D      2 type
18223:D      2     byte = 0..255;
18224:S
18225:D      2 var
18226:D      -2 2     cnt : gle_shortint;
18227:D      -4 2     button : gle_shortint;
18228:D      -4 2     status : packed record
18229:D      -4 2         case byte of
18230:D      -4 2             0 : (whole : gle_shortint);
18231:D      -4 2             1 : (part : packed record
18232:D      -4 2                 bit15,bit14,bit13,bit12,bit11,
18233:D      -4 2                 pen_down,
18234:D      -4 2                 new_cursor,

```

```

18235:D      -4 2     proximity,
18236:D      -4 2     softkey,
18237:D      -4 2     srq,
18238:D      -4 2     error,
18239:D      -4 2     ready,
18240:D      -4 2     init,
18241:D      -4 2     point_ready,
18242:D      -4 2     bit1,
18243:D      -4 2     bit0 : boolean;
18244:D      -4 2     end;
18245:D      -6 2     end;
18246:S
18247:C      2 begin
18248:C      2     with gcbl^,
18249:C      3     ascll_buffer_ptr(gcbl^.device_buf)^ do
18250:C      3     begin
***WARNING: (line18251): 'ADDR' of a constant may not be supported on other implementations
18251:C      3     add_char_data ( gcbl, 2, addr('OR') );
18252:C      3     hpgl_flush_buffer ( gcbl );
18253:C      3     call (io_read, iocb, device_buf);
18254:C      3     info1 := gle_read_integer (current,addr(data[1]),cnt);
18255:C      3     info2 := gle_read_integer (current,addr(data[cnt+1]),cnt);
18256:C      3     current := 0;
18257:S
18258:C      3     input_cpx := info1;
18259:C      3     input_cpy := info2;
18260:S
***WARNING: (line18261): 'ADDR' of a constant may not be supported on other implementations
18261:C      3     add_char_data ( gcbl, 2, addr('OS') );
18262:C      3     hpgl_flush_buffer ( gcbl );
18263:C      3     call (io_read, iocb, device_buf);
18264:S
18265:C      3     status.whole := gle_read_integer (current,addr(data[1]),cnt);
18266:C      3     current := 0;
18267:S
18268:C      3     if status.part.point_ready then
18269:C      4         info3 := -1
18270:C      4     else
18271:C      4         if status.part.pen_down then
18272:C      5             info3 := 1
18273:C      5         else
18274:C      5             info3 := 0;
18275:C      5     end;
18276:C      2 end;
18277:S
18278:D      1 procedure hpgl_start_digitize ( gcbl : graphics_input_control_block_ptr );
18279:S
18280:C      2 begin
18281:C      2     with gcbl^ do
18282:C      3     begin
***WARNING: (line18283): 'ADDR' of a constant may not be supported on other implementations
18283:C      3     add_char_data ( gcbl, 8, addr('PU;SG;DP'));
18284:C      3     hpgl_flush_buffer ( gcbl );
18285:C      3     end;
18286:S
18287:S
18288:D      1 procedure hpgl_get_digitize ( gcbl : graphics_input_control_block_ptr );
18289:S
18290:D      2 var
18291:D      -4 2     cnt,cnt : gle_shortint;

```



```

18292:S
18293:C      2 begin
18294:C      2   with gcbl^,
18295:C      3     ascii_buffer_ptr(gcbl^.device_buf)^ do
18296:C      3     begin
***WARNING: (line18297): 'ADDR' of a constant may not be supported on other implementations
18297:C      3       add_char_data ( gcbl, 2, addr('OD') );
18298:C      3       hpgl_flush_buffer ( gcbl );
18299:C      3       call (io_read, iocb, device_buf);
18300:C      3       tcnt := 1;
18301:C      3       info1 := gle_read_integer (current,addr(data[1]),tcnt);
18302:C      3       cnt := cnt + tcnt;
18303:C      3       info2 := gle_read_integer (current,addr(data[cnt]),tcnt);
18304:C      3       cnt := cnt + tcnt;
18305:C      3       info3 := gle_read_integer (current,addr(data[cnt]),tcnt);
18306:C      3       current := 0;
***WARNING: (line18307): 'ADDR' of a constant may not be supported on other implementations
18307:C      3       add_char_data ( gcbl, 2, addr('DC') );
18308:C      3       hpgl_flush_buffer ( gcbl );
18309:C      3     end;
18310:C      2 end;
18311:S
18312:D      1 procedure hpgl_input_echo ( gcbl : graphics_input_control_block_ptr );
18313:S
18314:C      2 begin
18315:C      2   with gcbl^ do
18316:C      3     begin
18317:C      3       if info1 <> 0 then
18318:C      4         begin
***WARNING: (line18319): 'ADDR' of a constant may not be supported on other implementations
18319:C      4           add_char_data ( gcbl, 2, addr('BP') );
18320:C      4           hpgl_flush_buffer ( gcbl );
18321:C      4         end;
18322:C      3     end;
18323:C      2 end;
18324:S
18325:D      1 procedure gle_init_hpgl_input ( gcbl : graphics_input_control_block_ptr);
18326:S
18327:D      2 var
18328:D      2   saved_timeout : integer;
18329:D      -8   i : integer;
18330:S
18331:C      2 begin
18332:C      2   with gcbl^, ascii_buffer_ptr(device_buf)^ do
18333:C      3     begin
18334:C      3       try
18335:C      4         maximum := max_buffer;
18336:C      4         current := 0;
18337:C      4         error_return := 0;
18338:S
18339:C      4         call (io_inq_timeout, iocb, saved_timeout );
18340:C      4         call (io_set_timeout, iocb, 500 (ms) );
18341:C      4         { send command that all HPGL plotters can respond to }
18342:C      4         { if the command fails then the address does not match }
18343:C      4         { the device. }
***WARNING: (line18344): 'ADDR' of a constant may not be supported on other implementations
18344:C      4         add_char_data ( gcbl, 2, addr('OE') );
18345:C      4         hpgl_flush_buffer ( gcbl );
18346:C      4         call (io_read, iocb, device_buf);
18347:C      4         current := 0;

```

```

18348:S
18349:C      4   { if this point is reached then a valid HPGL device was found }
18350:S
18351:C      4   try { perform an output identify seq. Note a 9872A will fail }
***WARNING: (line18352): 'ADDR' of a constant may not be supported on other implementations
18352:C      4   add_char_data ( gcbl, 10, addr('I1M30;DC;01') );
18353:C      5   hpgl_flush_buffer ( gcbl );
18354:C      5   call (io_read, iocb, device_buf);
18355:S
18356:C      5   for i := 1 to current do input_name[i] := data[i];
18357:C      5   for i := current+1 to 6 do input_name[i] := ' ';
18358:C      5   input_name_char_count := current;
18359:C      5   current := 0;
18360:C      5   recover
18361:C      5   if escapecode = -26 then
18362:C      6     begin
18363:C      6       input_name := '9872A';
18364:C      6       input_name_char_count := 5;
18365:C      6       current := 0;
18366:C      6     end
18367:C      6   else
18368:C      6     escape(escapecode);
18369:S
18370:C      4   sample      := hpgl_sample;
18371:C      4   start_digitize := hpgl_start_digitize;
18372:C      4   get_digitize   := hpgl_get_digitize;
18373:C      4   inq_plp2      := hpgl_get_input_plp2;
18374:C      4   input_echo     := hpgl_input_echo;
18375:C      4   input_escape1  := hpgl_input_escape1;
18376:C      4   input_escape0  := hpgl_input_escape0;
18377:S
18378:C      4   input_res_x := 40;
18379:C      4   input_res_y := 40;
18380:S
18381:C      4   input_handler_name := 'HPGL';
18382:C      4   input_handler_char_count := 4;
18383:S
18384:C      4   hpgl_get_input_hard_clip ( gcbl );
18385:C      4   input_min_x := info1;
18386:C      4   input_max_x := info2;
18387:C      4   input_min_y := info3;
18388:C      4   input_max_y := info4;
18389:S
18390:C      4   input_cpx := input_min_x;
18391:C      4   input_cpy := input_min_y;
18392:S
18393:C      4   recover
18394:C      4   if escapecode = -26 then error_return := 1
18395:C      5   else escape(escapecode); { ignor io errors }
18396:S
18397:C      3   try
18398:C      4     call (io_set_timeout, iocb, saved_timeout );
18399:C      4   recover
18400:C      4   if escapecode = -26 then error_return := 1
18401:C      5   else escape(escapecode); { ignor io errors }
18402:C      3   end;
18403:C      2 end;
18404:S
18405:S
18406:C      1 end. { hpgl_input }

```


18407:S

No errors. 25 warnings.

**** Nonstandard language features enabled ****

GLE_HPIB

Description

GLE_HPIB contains I/O routines which interface the graphics library to the system HP-IB I/O drivers.

Usage

Used with HPGL displays or locators connected via HP-IB.

Requirements

GLE_TYPES, GENERAL_0, IODECLARATIONS, SYSGLOBALS, IOCOMASM, and GLE_UTLS.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S
2:D
3:D 0 { Graphics Low End }
4:D 0 { }
5:D 0 { Module = GLE_HPIB }
6:D 0 { Programmer = BJS }
7:D 0 { Date = 10-10-82 }
8:D 0 { }
9:D 0 { Purpose: To provide IO routines for ascii device handlers. }
10:S
11:D 0 { Rev history }
12:D 0 { Created - 10-10-82 }
13:D 0 { Modified - 12-12-83 BDS -- Brought needed general_1 and hpiib_1 }
14:D 0 { routines in-line. }
15:S
16:S {
17:S { (c) Copyright Hewlett-Packard Company, 1983.
18:S { All rights are reserved. Copying or other
19:S { reproduction of this program except for archival
20:S { purposes is prohibited without the prior
21:S { written consent of Hewlett-Packard Company.
22:S
23:S
24:S
25:S
26:S
27:S
28:S
29:S
30:S
31:S
32:D 0
33:S
34:D 0 $SEARCH 'GLE_TYPES'.
35:D 0 'GLE_UTILS'$
36:D 0 $modcls$
37:D 0 $include 'OPTIONS'$ ( ***** COMPILER OPTIONS ***** )
38:S
39:D 0 { This include file specifies range checking, debug and other compiler
40:D 0 options for the graphics library }
41:D 0 $debug OFF$
42:D 0 $range OFF$
43:D 0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'S
44:D 0 $FLOAT_HDW TESTS
45:S
46:S
47:S
48:S
49:D 0 $include 'OPTIONS'$ ( ***** COMPILER OPTIONS ***** )
50:D 0 $LINENUM 10000$
10000:D
10001:D
10002:D
10003:D
10004:D 1 import gle_types,
10005:D 1 general_0,
10006:D 1 locomas;
10007:D
10008:D 1 export
10009:D 1 type

```

```

10010:D 1 hpiib_iocb_ptr = ^ hpiib_iocb;
10011:D 1 hpiib_iocb = record
10012:D 1 device_addr : anyptr;
10013:D 1 name_size : gle_shortint;
10014:D 1 address : integer;
10015:D 1 select_code : integer;
10016:D 1 error : integer;
10017:D 1 end;
10018:D 1 timeoutrec = record { tttt JS 8/3/83 }
10019:D 1 counter : integer; { tttt JS 8/3/83 }
10020:D 1 firsttime : boolean; { tttt JS 8/3/83 }
10021:D 1 end; { tttt JS 8/3/83 }
10022:S
10023:D 1 procedure hpiib_init ( anyvar iocb_ptr : anyptr );
10024:D 1 procedure hpiib_inq_timeout ( anyvar iocb_ptr : anyptr; var value : integer );
10025:D 1 procedure hpiib_set_timeout ( anyvar iocb_ptr : anyptr; value : integer );
10026:D 1 procedure hpiib_write ( anyvar iocb_ptr, data_ptr : anyptr );
10027:D 1 procedure hpiib_read ( anyvar iocb_ptr, data_ptr : anyptr );
10028:D 1 procedure hpiib_term ( anyvar iocb_ptr : anyptr );
10029:S
10030:D 1 implement
10031:S
10032:D 1 import
10033:D 1 { declarations,
10034:D 1 { general_1,
10035:D 1 { hpiib_1,
10036:D 1 gle_utils;
10037:S
10038:D 1 { The following types must match types declared in GLE_HPGL, and GLE_HPGLI }
10039:D 1 type
10040:D 1 { ascii_buffer_ptr = ^ascii_buffer;
10041:S
10042:D 1 { ascii_buffer = packed record
10043:D 1 { maximum : integer;
10044:D 1 { current : integer;
10045:D 1 { data : packed array [1..32767] of char;
10046:D 1 { end;
10047:S
10048:D 1 { (***** general_1 *****)
10049:S
10050:D 1 { (*** general_1 ***)
10051:S
10052:S {The following procedures have been brought in-line to avoid importing
10053:S {general_1 and hpiib_1. NOTE: These routines must be duplicates of those
10054:S {found in general_1 and hpiib_1 and therefore must reflect any modifications
10055:D 1 {to those modules.
10056:S
10057:S
10058:S
10059:D 1 FUNCTION timerexists: boolean; external; { tttt JS 8/3/83 }
10060:D
10061:D 1 FUNCTION timed_out(var rec: timeoutrec): boolean; external; { tttt JS 8/3/83 }
10062:S
10063:S
10064:D 1 PROCEDURE ioreset ( select_code : type_isc);
10065:C 2 BEGIN
10066:C 2 WITH isc_table[select_code] DO
10067:C 3 CALL(io_drv_ptr^.iod_init,
10068:C 3 io_tmp_ptr);
10069:C 2 END; { of ioreset }

```



```

10070:S
10071:S
10072:D 1  PROCEDURE writechar ( select_code : type_isc ;
10073:D 2  value : CHAR );
10074:C 2  BEGIN
10075:C 3  WITH isc_table[select_code] DO
10076:C 3  CALL (io_drv_ptr^.iod_wib,
10077:C 3  io_tmp_ptr,
10078:C 3  value);
10079:C 2  END; ( of writechar )
10080:S
10081:D -8 1  PROCEDURE set_timeout ( select_code : type_isc ;
10082:D 2  time : REAL ( in seconds ) );
10083:C 2  BEGIN
10084:C 2  IF time>8191 ( 4 byte timeout - 1 byte left for shifts )
10085:C 3  THEN BEGIN
10086:C 3  { error }
10087:C 3  io_escape(ioe_bad_tmo,select_code);
10088:C 3  END; ( of IF )
10089:C 3  IF (time>0) AND (time<0.001)
10090:C 3  THEN BEGIN
10091:C 3  { error }
10092:C 3  io_escape(ioe_bad_tmo,select_code);
10093:C 3  END; ( of IF )
10094:S
10095:C 2  WITH isc_table[select_code] DO BEGIN
10096:S
10097:C 3  { the table entry used by drivers is in milliseconds }
10098:C 3  user_time:=ROUND(time*1000);
10099:S
10100:C 3  IF io_tmp_ptr <> NIL THEN io_tmp_ptr^.timeout := user_time;
10101:S
10102:C 3  END; ( of WITH DO BEGIN )
10103:S
10104:C 2  END; ( of set_timeout )
10105:D
10106:D 1  (*** hpib_1 ***)
10107:S
10108:D 1  PROCEDURE send_command( select_code : type_isc ;
10109:D 2  command : CHAR );
10110:C 2  BEGIN
10111:C 3  WITH isc_table[select_code] DO
10112:C 3  CALL ( io_drv_ptr^.iod_send,
10113:C 3  io_tmp_ptr,
10114:C 3  command);
10115:C 2  END; ( of send_command )
10116:S
10117:S
10118:S
10119:D 1  FUNCTION my_address ( select_code : type_isc)
10120:D 2  : type_hpib_addr ;
10121:C 2  BEGIN
10122:C 3  IF isc_table[select_code].io_tmp_ptr <> NIL
10123:C 3  THEN BEGIN
10124:C 3  WITH isc_table[select_code].io_tmp_ptr^ DO
10125:C 4  IF addressed <> -1
10126:C 4  THEN BEGIN
10127:C 4  my_address:=addressed;
10128:C 4  END
10129:C 3  ELSE BEGIN

```

```

10130:C 5  { error }
10131:C 5  io_escape(ioe_not_hpib,select_code);
10132:C 5  END; ( of IF addressed )
10133:C 3  END
10134:C 3  ELSE BEGIN
10135:C 3  { error }
10136:C 3  io_escape(ioe_not_hpib,select_code);
10137:C 3  END; ( of IF io_tmp_ptr )
10138:C 2  END; ( of my_address )
10139:S
10140:S
10141:S
10142:D 1  FUNCTION active_controller
10143:D 2  ( select_code : type_isc)
10144:D 2  : BOOLEAN;
10145:C 2  BEGIN
10146:C 3  IF isc_table[select_code].card_type=hpib_card
10147:C 3  THEN BEGIN
10148:C 3  active_controller:=bit_set(iostatus(select_code,3),6);
10149:C 3  END
10150:C 3  ELSE BEGIN
10151:C 3  active_controller := TRUE;
10152:C 3  END; ( of IF )
10153:C 2  END; ( of active_controller )
10154:S
10155:S
10156:S
10157:D 1  (***** this function is not used *****)
10158:D 1  {FUNCTION system_controller
10159:D 2  ( select_code : type_isc)
10160:D 2  : BOOLEAN;
10161:S
10162:S
10163:S
10164:S
10165:S
10166:S
10167:S
10168:S
10169:D 1  BEGIN
10170:S
10171:S
10172:S
10173:D 1  (***** this function is not used *****)
10174:D 1  {FUNCTION end_set
10175:D 2  ( select_code : type_isc )
10176:D 2  : BOOLEAN;
10177:S
10178:S
10179:S
10180:S
10181:S
10182:S
10183:D 1  VAR mybool : BOOLEAN;
10184:S
10185:S

```



```

10186:D 1 $PAGE$
10187:S 1
10188:D 1 FUNCTION addr_to_talk( device : type_device)
10189:D 2 : type_isc;
10190:D -2 2 VAR io_isc : type_isc;
10191:D -6 2 timer : INTEGER;
10192:D -12 2 hpibrec: timeoutrec; (tttt JS 8/3/83)
10193:S 2
10194:S 2 BEGIN
10195:S 2
10196:C 2 IF device>iomaxisc
10197:C 3 THEN BEGIN
10198:C 3 io_isc:=device DIV 100;
10199:S 3
10200:C 3 WITH isc_table[io_isc] DO BEGIN
10201:S 4
10202:C 4 IF io_tmp_ptr <> NIL
10203:C 5 THEN BEGIN
10204:S 5
10205:C 5 { set up user timeout - in case system drivers changed it }
10206:C 5 io_tmp_ptr^.timeout:=user_time;
10207:S 5
10208:C 5 IF io_tmp_ptr^.addressed <> -1
10209:C 6 THEN BEGIN
10210:C 6 IF ( card_type <> hpib_card ) AND
10211:C 7 ( device MOD 100 > 31 )
10212:C 7 THEN io_escape(ioe_misc,io_isc);
10213:C 6 send_command(io_isc,CHR(talk_constant+(device MOD 100)));
10214:C 6 send_command(io_isc,'?');
10215:C 6 send_command(io_isc,CHR(my_address(io_isc)+listen_constant));
10216:C 6 END;
10217:C 6 ELSE BEGIN
10218:C 6 ( error )
10219:C 6 io_escape(ioe_not_hpib,io_isc);
10220:C 6 END; { of IF }
10221:C 5 END
10222:C 5 ELSE BEGIN
10223:C 5 END; { of IF }
10224:C 4 END; { of WITH DO BEGIN }
10225:S 3
10226:C 3 ELSE BEGIN
10227:C 3 io_isc:=device;
10228:C 3
10229:C 3 WITH isc_table[io_isc] DO BEGIN
10230:S 4
10231:C 4 { set up user timeout - in case system drivers changed it }
10232:C 4 IF io_tmp_ptr <> NIL THEN io_tmp_ptr^.timeout:=user_time;
10233:S 4
10234:C 4 IF card_type=hpib_card THEN
10235:C 5 BEGIN
10236:C 5 IF NOT active_controller(io_isc)
10237:C 6 THEN BEGIN
10238:C 6 { if non controller wait until listener }
10239:C 6 IF user_time = 0
10240:C 7 THEN BEGIN
10241:C 7 REPEAT
10242:C 8 { wait forever }
10243:C 8 UNTIL bit_set(iostatus(io_isc,6),10);
10244:C 7 END
10245:C 7 ELSE BEGIN

```

```

10246:C 7 { wait for timeout value }
10247:C 7 IF timereexists THEN BEGIN (tttt JS 8/3/83)
10248:C 8 hpibrec.firsttime:=true; (tttt JS 8/3/83)
10249:C 8 hpibrec.counter:=user_time; (tttt JS 8/3/83)
10250:C 8 REPEAT (tttt JS 8/3/83)
10251:C 9 UNTIL timed_out(hpibrec) OR (tttt JS 8/3/83)
10252:C 9 bit_set(iostatus(io_isc,6),10); (tttt JS 8/3/83)
10253:C 8 END (tttt JS 8/3/83)
10254:C 8 ELSE BEGIN
10255:C 8 timer:=user_time*3;
10256:C 8 REPEAT
10257:C 9 timer:=timer-1;
10258:C 9 UNTIL ( timer = 0 ) OR
10259:C 9 ( bit_set(iostatus(io_isc,6),10) );
10260:C 8 END; (tttt JS 8/3/83)
10261:C 8 IF NOT bit_set(iostatus(io_isc,6),10)
10262:C 8 THEN io_escape(ioe_timeout,io_isc);
10263:C 7 END; { of IF user_time=0 }
10264:C 6 END; { of IF }
10265:C 5 END; { of IF card_type = hpib_card }
10266:C 4 END; { of WITH DO BEGIN }
10267:C 3 END; { of IF }
10268:S 2
10269:C 2 addr_to_talk:=io_isc; { return select code }
10270:S 2
10271:C 2 END; { of addr_to_talk }
10272:S 2

```



```

10273:0      1  $PAGE$
10274:S
10275:D      1  FUNCTION addr_to_listen
10276:0      2      ( device      : type_device)
10277:0      2      ( type_isc;
10278:D      -2 2  VAR io_isc : type_isc;
10278:D      -6 2  timer  : INTEGER;
10280:D      -12 2 hpbrec: timeoutrec;
10281:S
10282:C      2  BEGIN
10283:S
10284:C      2      IF device>iomaxisc
10285:C      3      THEN BEGIN
10286:C      3          io_isc:=device DIV 100;
10287:S
10288:C      3          WITH isc_table[io_isc] DO BEGIN
10289:S
10290:C      4              IF io_tmp_ptr <> NIL
10291:C      5              THEN BEGIN
10292:S
10293:C      5                  { set up user timeout - in case system drivers changed it }
10294:C      5                  io_tmp_ptr^.timeout:=user_time;
10295:S
10296:C      5                  IF io_tmp_ptr^.addressed <> -1
10297:C      6                  THEN BEGIN
10298:C      6                      IF ( card_type <> hpib_card ) AND
10299:C      7                      ( device MOD 100 > 31 )
10300:C      7                      THEN io_escape(io_isc,io_isc);
10301:C      6                      send_command(io_isc,CHR(my_address(io_isc)+talk_constant));
10302:C      6                      send_command(io_isc,'?');
10303:C      6                      send_command(io_isc,CHR(listen_constant+(device MOD 100)));
10304:C      6                      ENO
10305:C      6                      ELSE BEGIN
10306:C      6                          { error }
10307:C      6                          io_escape(io_not_hpib,io_isc);
10308:C      6                      END; { of IF }
10309:C      5                  END
10310:C      5                  ELSE BEGIN
10311:C      5                      END; { of IF }
10312:C      4                  END; { of WITH DO BEGIN }
10313:C      3                  ENO
10314:C      3                  ELSE BEGIN
10315:C      3                      io_isc:=device;
10316:C      3                      WITH isc_table[io_isc] DO BEGIN
10317:C      3                          { set up user timeout - in case system drivers changed it }
10318:S
10319:C      4                          IF io_tmp_ptr <> NIL THEN io_tmp_ptr^.timeout:=user_time;
10320:C      4                          IF card_type=hpib_card THEN
10321:S
10322:C      4                              BEGIN
10323:C      5                                  IF NOT active_controller(io_isc)
10324:C      5                                  THEN BEGIN
10325:C      6                                      { if non controller wait until talker }
10326:C      6                                      IF user_time = 0
10327:C      6                                      THEN BEGIN
10328:C      7                                          REPEAT
10329:C      7                                              { wait forever }
10330:C      8                                              UNTIL bit_set(iostatus(io_isc,6),9);
10331:C      8                                          ENO
10332:C      7                                          REPEAT
10332:C      7                                          UNTIL bit_set(iostatus(io_isc,6),9);
10332:C      7                                          ENO

```

```

10333:C      7      ELSE BEGIN
10334:C      7          { wait for timeout value }
10335:C      7          IF timerexists THEN BEGIN
10336:C      8              hpbrec.firsttime:=true;
10337:C      8              hpbrec.counter:=user_time;
10338:C      8              REPEAT
10339:C      8                  UNTIL timed_out(hpbrec) OR
10340:C      9                  bit_set(iostatus(io_isc,6), 9);
10341:C      8              ENO
10342:C      8              ELSE BEGIN
10343:C      8                  timer:=user_time*3;
10344:C      8                  REPEAT
10345:C      9                      timer:=timer-1;
10346:C      9                      UNTIL ( timer = 0 ) OR
10347:C      9                      ( bit_set(iostatus(io_isc,6),9) );
10348:C      8                  ENO;
10349:C      7                  IF NOT bit_set(iostatus(io_isc,6),9)
10350:C      7                  THEN io_escape(io_timeout,io_isc);
10351:C      7                  ENO; { of IF user_time=0 }
10352:C      6                  ENO; { of IF }
10353:C      5                  END; { of IF card_type = hpib_card }
10354:C      4                  END; { of WITH DO BEGIN }
10355:C      3                  END; { of IF }
10356:S
10357:C      2      addr_to_listen:=io_isc;
10358:S
10359:C      2      END; { of addr_to_listen }
10360:S

```



```

10361:D      1  $PAGE$
10362:S
10363:S
10364:S      ( set to talk exists because of HPIB_2/HPIB_3 -
10365:S      those routines are intended to be the controller
10366:S      ( active ) and should not wait for the card to be
10367:S      addressed as talker. addr_to_talk is used by
10368:S      data transfer routines. set_to_talk is used by
10369:D      1  bus control routines. )
10370:S
10371:S
10372:S
10373:S
10374:D      1  {***** this function is never used *****}
10375:S      {FUNCTION set_to_talk ( device      : type_device)
10376:S      ( device      : type_device)
10377:S      VAR io_isc : type_isc;
10378:S      BEGIN
10379:S
10380:S          IF device>iomaxisc
10381:S          THEN BEGIN
10382:S              io_isc:=addr_to_talk(device);
10383:S          END
10384:S          ELSE BEGIN
10385:S              io_isc:=device;
10386:S          END
10387:S          WITH isc_table[io_isc] DO BEGIN
10388:S
10389:S              set up user timeout - in case system drivers changed it
10390:S              IF io_tmp_ptr <> NIL THEN io_tmp_ptr.timeout:=user_time;
10391:S
10392:S              IF card_type=hpib_card THEN
10393:S              BEGIN
10394:S                  IF NOT active_controller(io_isc)
10395:S                  THEN BEGIN
10396:S
10397:S                      io_escape(ioe_not_act,io_isc);
10398:S
10399:S                  END; of IF
10400:S              END; of IF card type = hpib_card
10401:S              END; of WITH DO BEGIN
10402:S              END; of IF
10403:S
10404:S              set_to_talk:=io_isc; return select code
10405:S
10406:D      1  END; of set_to_talk )
10407:S

```

```

10408:D      1  $PAGE$
10409:S
10410:S
10411:S      ( set to listen exists because of HPIB_2/HPIB_3 -
10412:S      those routines are intended to be the controller
10413:S      ( active ) and should not wait for the card to be
10414:S      addressed as listener. addr_to_listen is used by
10415:S      data transfer routines. set_to_listen is used by
10416:D      1  bus control routines. )
10417:S
10418:S
10419:S
10420:S
10421:S
10422:D      1  {***** this function is never used *****}
10423:S      {FUNCTION set_to_listen
10424:S      ( device      : type_device)
10425:S      ( device      : type_device)
10426:S      VAR io_isc : type_isc;
10427:S      timer : INTEGER;
10428:S      BEGIN
10429:S
10430:S          IF device>iomaxisc
10431:S          THEN BEGIN
10432:S              io_isc:=addr_to_listen(device);
10433:S          END
10434:S          ELSE BEGIN
10435:S              io_isc:=device;
10436:S          END
10437:S          WITH isc_table[io_isc] DO BEGIN
10438:S
10439:S              set up user timeout - in case system drivers changed it
10440:S              IF io_tmp_ptr <> NIL THEN io_tmp_ptr.timeout:=user_time;
10441:S
10442:S              IF card_type=hpib_card THEN
10443:S              BEGIN
10444:S                  IF NOT active_controller(io_isc)
10445:S                  THEN BEGIN
10446:S
10447:S                      io_escape(ioe_not_act,io_isc);
10448:S
10449:S                  END; of IF
10450:S              END; of IF card type = hpib_card
10451:S              END; of WITH DO BEGIN
10452:S              END; of IF
10453:S
10454:S              set_to_listen:=io_isc;
10455:S
10456:D      1  END; of set_to_listen )
10457:S
10458:S
10459:S
10460:S
10461:S
10462:S
10463:S
10464:D      1  {*****}
10465:S
10466:S
10467:D      1  procedure hpib_inq_timeout ( anyvar iocb_ptr : anyptr; var value : integer );

```



```

10468:S      2 begin
10469:C      2   with hpib_iocb_ptr(iocb_ptr)^ do
10470:C      3   begin
10471:C      3     value := isc_table[select_code].user_time;
10472:C      3   end;
10473:C      2 end;
10474:C      2 end;
10475:S      1 procedure hpib_set_timeout ( anyvar iocb_ptr : anyptr; value : integer );
10476:D      2 begin
10477:C      2   with hpib_iocb_ptr(iocb_ptr)^ do
10478:C      3   set_timeout (select_code,value/1000);
10479:C      2 end;
10480:C      2 end;
10481:C      2 end;
10482:S      1 procedure hpib_init ( anyvar iocb_ptr : anyptr );
10483:D      2 var
10484:S      2   cnt : gle_shortint;
10485:D      -2 2   cnt := 1;
10486:D      2 begin
10487:C      2   with hpib_iocb_ptr(iocb_ptr)^ do
10488:C      3   begin
10489:C      3     error := 1;
10490:C      3     try
10491:C      4     address := gle_read_integer ( name_size, device_addr, cnt );
10492:C      4     select_code := address div 100;
10493:C      4     if (select_code >= minrealisc) and
10494:C      5     (select_code <= maxrealisc) then
10495:C      5     begin
10496:C      5       ioreset (select_code);
10497:C      5       error := 0;
10498:C      5     end;
10499:C      5     recover
10500:C      4     { error is set, ignore range and io escapes }
10501:C      4     if (escapecode <> -8) and (escapecode <> -26) then escape(escapecode);
10502:C      4   end;
10503:C      2 end;
10504:C      2 end;
10505:S      1 procedure hpib_write ( anyvar iocb_ptr, data_ptr : anyptr );
10506:D      2 var
10507:S      2   i : integer;
10508:D      -4 2   i := 1;
10509:D      -6 2   io_isc : type_isc;
10510:D      2 begin
10511:C      2   with hpib_iocb_ptr(iocb_ptr)^,ascii_buffer_ptr(data_ptr)^ do
10512:C      3   begin
10513:C      3     io_isc := addr_to_listen(address);
10514:C      3     with isc_table[io_isc].io_drv_ptr^, isc_table[io_isc] do
10515:C      4     begin
10516:C      4       for i := 1 to current do
10517:C      5       call (iod_wtb, io_tmp_ptr, data[i]);
10518:C      5       writechar(io_isc,io_carriage_rtn);
10519:C      5       writechar(io_isc,io_line_feed);
10520:C      5     end;
10521:C      4     current := 0;
10522:C      4   end;
10523:C      2 end;
10524:C      2 end;
10525:S      2 end;
10526:C      2 end;
10527:S

```

```

10528:D      1 procedure hpib_read ( anyvar iocb_ptr, data_ptr : anyptr );
10529:S      2 var
10530:D      -4 2   i : integer;
10531:D      -6 2   io_isc : type_isc;
10532:D      2 begin
10533:C      2   with hpib_iocb_ptr(iocb_ptr)^,ascii_buffer_ptr(data_ptr)^ do
10534:C      3   begin
10535:C      3     io_isc := addr_to_talk ( address );
10536:C      3     with isc_table[io_isc].io_drv_ptr^, isc_table[io_isc] do
10537:C      4     begin
10538:C      4       i := 0;
10539:C      4       repeat
10540:C      5       i := i + 1;
10541:C      5       call (iod_rdb, io_tmp_ptr, data[i]);
10542:C      5       until { ( i >= maximum ) or ( data[i] = io_line_feed ) };
10543:C      5       if data[i] = io_line_feed then i := i - 1;
10544:C      5       if i <> 0 then if data[i] = io_carriage_rtn then i := i - 1;
10545:C      5       current := i;
10546:C      4     end;
10547:C      3   end;
10548:C      2 end;
10549:C      2 end;
10550:S      1 procedure hpib_term ( anyvar iocb_ptr : anyptr );
10551:D      2 begin
10552:C      2 end;
10553:S      1 end. { of module gle_hpib_io }
10554:D      1
10555:D      1 $LIST ON$
10556:D

```

No errors. No warnings.

***** Nonstandard language features enabled *****

GLE_KNOB

Description

GLE_KNOB provides a device handler for the built-in knob input device.

Requirements

GLE_TYPES, GLE_UTLS, and SYSGLOBALS.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      0 {
2:D      0 { Graphics Low End
3:D      0 {
4:D      0 { Module = GLE_KNOB_IN
5:D      0 { Programmer = 8JS
6:D      0 { Date = 11-05-82
7:D      0 {
8:D      0 {
9:D      0 { Purpose: To provide a device handler for the knob input device.
10:S     0 {
11:D     0 { Rev history
12:D     0 { Created - 11- 5-82
13:D     0 { Modified - 4-11-84 BY JWS -- remove unitio dependencies
14:S     0 {
15:S     ( (c) Copyright Hewlett-Packard Company, 1983.
16:S     All rights are reserved. Copying or other
17:S     reproduction of this program except for archival
18:S     purposes is prohibited without the prior
19:S     written consent of Hewlett-Packard Company.
20:S
21:S
22:S
23:S
24:S     RESTRICTED RIGHTS LEGEND
25:S
26:S     Use, duplication, or disclosure by the Government
27:S     is subject to restrictions as set forth in
28:S     paragraph (b) (3) (8) of the Rights in Technical
29:S     Data and Computer Software clause in
30:S     DAR 7-104.9(a).
31:D     0
32:S     HEWLETT-PACKARD COMPANY
33:S     Fort Collins, Colorado
34:D     0
35:D     0 $UCSD$
36:D     0 $SEARCH 'GLE_TYPES', 'GLE_UTILS'$
37:D     0 $modals$
38:D     0 $INCLUDE 'OPTIONS'$ ( ***** COMPILER OPTIONS ***** )
39:S     ( This include file specifies range checking, debug and other compiler
40:S     options for the graphics library
41:D     0 $debug OFF$
42:D     0 $range OFF$
43:D     0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
44:D     0 $FLOAT_HDW TEST$
45:S
46:S
47:S
48:S
49:D     0 $INCLUDE 'OPTIONS'$ ( ***** COMPILER OPTIONS ***** )
50:D     0 $LINENUM 18000$
18001:D  0
18002:D  0 module gle_knob_in;
18003:S  1
18004:D  1 import gle_types;
18005:S  1
18006:D  1 export
18007:D  1
18008:D  1 type
18009:D  1 knob_device_rec_ptr = ^knob_device_rec;

```

```

18010:D  1 knob_device_rec = packed record
18011:D  1 knob_type : gle_shortint; { 0 - unknown device }
18012:D  1 { 1 - 9816
18013:D  1 { 2 - 9826
18014:D  1 { 3 - 9836
18015:D  1 { 4 - 9836c
18016:D  1 echo_rate : gle_shortint;
18017:D  1 echo_mult : gle_shortint;
18018:D  1 last_but : integer;
18019:D  1 digitize_enabled : boolean;
18020:D  1 last_x : integer;
18021:D  1 last_y : integer;
18022:D  1 end;
18023:S  1
18024:D  1 procedure gle_init_knob_input ( gcbl : graphics_input_control_block_ptr );
18025:D  1
18026:D  1 implement
18027:S  1
18028:D  1 import gle_utils, sysglobals;
18029:S  1
18030:S  1
18031:D  1 procedure eot(fp: fibp); begin end; {do nothing}
18032:S  1
18033:D  1 procedure openfib(anyvar F: fib; unum: unitnum);
18034:C  2 begin
18035:C  2 if (unum=0) or (unum>maxunit) then ioresult := ord(ibadunit)
18036:C  3 else with F do
18037:C  4 begin
18038:C  4 fistextvar := false;
18039:C  4 funit := unum;
18040:C  4 froot := eot;
18041:C  4 call(unitable^[unum].dam, f, unum, openunit);
18042:C  4 end;
18043:C  2 end;
18044:S  1
18045:D  1 procedure gunitread ( u: integer; buf: charptr; len: integer);
18046:D  2 var f: file;
18047:D  2 var r: amrequesttype;
18048:C  2 begin with unitable[u] do
18049:C  3 begin
18050:C  3 openfib(f, u);
18051:C  3 if ioresult = ord(inoerror) then
18052:C  4 begin
18053:C  4 r := readbytes;
18054:C  4 call(tm, addr(f), r, buf^, len, 0);
18055:C  4 end;
18056:C  3 end;
18057:C  2 end;
18058:S  1
18059:D  1 function gunitbusy ( u: integer): boolean;
18060:D  2 var f: file;
18061:D  2 begin with unitable[u] do
18062:C  3 begin
18063:C  3 gunitbusy := true;
18064:C  3 openfib(f, u);
18065:C  3 if ioresult = ord(inoerror) then
18066:C  4 begin
18067:C  4 call(tm, addr(f), unitstatus, f, 0, 0);
18068:C  4 gunitbusy := fibp(addr(f))^fbusy;
18069:C  4 end;

```



```

18070:C      3   end;
18071:C      2 end;
18072:S
18073:S
18074:S
18075:D
18076:S
18077:C      1 procedure knob_dummy ( gcbl : graphics_input_control_block_ptr );
18078:C      2 begin
18079:C      2 end;
18080:D
18081:S
18082:C      2 begin
18083:C      2   with gcbl^,knob_device_rec_ptr(dev_dep_stuff)^ do
18084:C      3     case knob_type of
18085:C      4       0 : begin
18086:C      4         info1 := input_min_x; { input_xxxx must be set by config }
18087:C      4         info2 := input_max_x;
18088:C      4         info3 := input_min_y;
18089:C      4         info4 := input_max_y;
18090:C      4       end;
18091:C      4       1,2 : begin
18092:C      4         info1 := 0;
18093:C      4         info2 := 389;
18094:C      4         info3 := 0;
18095:C      4         info4 := 299;
18096:C      4       end;
18097:C      4       3,4 : begin
18098:C      4         info1 := 0;
18099:C      4         info2 := 511;
18100:C      4         info3 := 0;
18101:C      4         info4 := 389;
18102:C      4       end;
18103:C      4     end; { case }
18104:C      2 end;
18105:S
18106:D
18107:S
18108:D
18109:D      2 var
18110:D      2   ch : char;
18111:D      2   mycharptr: charptr;
18112:D      2   x_adj;
18113:D      2   y_adj;
18114:D      2   rate : integer;
18115:D      2   command_inprogress: char;
18116:D      2   interruptlevel: integer;
18117:C      2 begin
18118:C      2   mycharptr:=addr(ch);
18119:C      2   with gcbl^,knob_device_rec_ptr(dev_dep_stuff)^ do
18120:C      3     begin
18121:C      4       if last_but < 0 then info3 := -1
18122:C      4       else info3 := 0;
18123:C      4       if gunitbusy(2) then
18124:C      4         ( if no keys in type ahead buffer )
18125:C      4         begin
18126:C      4           { just return last position }
18127:C      4           info1 := input_cpx;
18128:C      4           info2 := input_cpy;
18129:C      4         end

```

```

18130:C      4     else
18131:C      4       begin
18132:C      4         gunitread(2,mycharptr,1);
18133:C      4         x_adj := 0;
18134:C      4         y_adj := 0;
18135:C      4         rate := 0;
18136:C      4         case ord(ch) of
18137:C      5           8 : x_adj := -echo_rate; { left arrow }
18138:C      5           28 : x_adj := +echo_rate; { right arrow }
18139:C      5           10 : y_adj := -echo_rate; { down arrow }
18140:C      5           31 : y_adj := +echo_rate; { up arrow }
18141:C      5           49,33 : rate := 1; { number 1 }
18142:C      5           50,64 : rate := 2; { ... 2 }
18143:C      5           51,35 : rate := 3;
18144:C      5           52,36 : rate := 4;
18145:C      5           53,37 : rate := 5;
18146:C      5           54,34 : rate := 6;
18147:C      5           55,38 : rate := 7;
18148:C      5           56,42 : rate := 8;
18149:C      5           57,40 : rate := 9; { number 9 }
18150:C      5         otherwise
18151:C      5           begin
18152:C      5             if ch = chr(13) then ch := ' ';
18153:C      5             if digitize_enabled then
18154:C      6               begin
18155:C      6                 if (last_but = 0) then
18156:C      7                   begin
18157:C      7                     last_but := -ord(ch);
18158:C      7                     info3 := -1;
18159:C      7                     last_x := input_cpx;
18160:C      7                     last_y := input_cpy;
18161:C      7                   end
18162:C      7                 end
18163:C      6             else
18164:C      6               info3 := ord(ch);
18165:C      5             end;
18166:C      5           end; { of case }
18167:C      5         end;
18168:S
18169:C      4         { ck for new rate }
18170:C      4         if rate > 0 then
18171:C      5           echo_rate := (echo_mult * (rate-1) + 1);
18172:S
18173:C      4         { calc new x and y }
18174:C      4         input_cpx := gle_shortint_max(input_min_x,gle_shortint_min(input_max_x,input_cpx+x_adj));
18175:C      4         input_cpy := gle_shortint_max(input_min_y,gle_shortint_min(input_max_y,input_cpy+y_adj));
18176:S
18177:C      4         info1 := input_cpx;
18178:C      4         info2 := input_cpy;
18179:C      4       end;
18180:C      2 end;
18181:C      2 end;
18182:S
18183:D
18184:S
18185:D
18186:D      2 begin
18187:D      2   with gcbl^,knob_device_rec_ptr(dev_dep_stuff)^ do
18188:D      3     begin
18189:D      3       digitize_enabled := true;

```



```

18190:C      3      last_but := 0;
18191:C      3      end;
18192:C      2 end;
18193:S
18194:D      1 procedure knob_get_digitize ( gcbi : graphics_input_control_block_ptr );
18195:S
18196:C      2 begin
18197:C      3      with gcbi^, knob_device_rec_ptr(dev_dep_stuff)^ do
18198:C      3      begin
18199:C      3          digitize_enabled := false;
18200:C      3          info1 := last_x;
18201:C      3          info2 := last_y;
18202:C      3          info3 := abs(last_but);
18203:C      3          last_but := 0;
18204:C      3      end
18205:C      3 end;
18206:S
18207:D      1 procedure knob_input_echo ( gcbi : graphics_input_control_block_ptr );
18208:S
18209:C      2 begin
18210:C      3      if gcbi^.info1 <> 0 then write(#G);
18211:C      2 end;
18212:S
18213:D      1 procedure gle_init_knob_input ( gcbi : graphics_input_control_block_ptr);
18214:S
18215:C      2 begin
18216:C      3      with gcbi^, knob_device_rec_ptr(dev_dep_stuff)^ do
18217:C      3      begin
18218:C      3          input_handler_name := 'KNOB ';
18219:C      3          input_handler_char_count := 4;
18220:C      3
18221:C      3          sample := knob_sample;
18222:C      3          start_digitize := knob_start_digitize;
18223:C      3          get_digitize := knob_get_digitize;
18224:C      3          input_echo := knob_input_echo;
18225:C      3          input_escape1 := knob_dummy;
18226:C      3          input_escape0 := knob_dummy;
18227:C      3          inq_plp2 := knob_get_input_plp2;
18228:C      3
18229:C      3          case knob_type of
18230:C      4              1 : begin
18231:C      4                  input_name := '9816A ';
18232:C      4                  input_name_char_count := 5;
18233:C      4                  input_res_x := 2.375; ( 168mm X 126mm )
18234:C      4                  input_res_y := 2.37301587301587;
18235:C      4                  end;
18236:C      4              2 : begin
18237:C      4                  input_name := '9826A ';
18238:C      4                  input_name_char_count := 5;
18239:C      4                  input_res_x := 3.325; ( 120mm X 80mm )
18240:C      4                  input_res_y := 3.32222222222222;
18241:C      4                  end;
18242:C      4              3 : begin
18243:C      4                  input_name := '9836A ';
18244:C      4                  input_name_char_count := 5;
18245:C      4                  input_res_x := 2.43333333333333; ( 210mm X 160mm )
18246:C      4                  input_res_y := 2.43125;
18247:C      4                  end;
18248:C      4              4 : begin
18249:C      4                  input_name := '9836C ';

```

```

18250:C      4          input_name_char_count := 5;
18251:C      4          input_res_x := 2.35483870967742; ( 217mm X 163mm )
18252:C      4          input_res_y := 2.38650306748466;
18253:C      4      end;
18254:C      4      otherwise ;
18255:C      4      end;
18256:C      3
18257:C      3      input_cpx := info1; ( setup init input values )
18258:C      3      input_cpy := info2;
18259:C      3      last_x := info1;
18260:C      3      last_y := info2;
18261:C      3      last_but := 0;
18262:S
18263:C      3      knob_get_input_plp2 ( gcbi );
18264:C      3      input_min_x := info1;
18265:C      3      input_max_x := info2;
18266:C      3      input_min_y := info3;
18267:C      3      input_max_y := info4;
18268:C      3
18269:C      3      digitize_enabled := false;
18270:C      3      echo_rate := 1;
18271:C      3      error_return := 0;
18272:C      3      end;
18273:C      2 end;
18274:S
18275:C      1 end. ( knob_input )
18276:S
18277:D      1 $LIST ON$

```

No errors. No warnings.

***** Nonstandard language features enabled *****

GLE_RGL

Description

GLE_RGL provides high-level device handler routines for raster devices.

Requirements

GLE_TYPES, SYSGLOBALS, SYSDEVS, GLE_ARAS_OUT, GLE_STEXT, GLE_ASTEXT, GLE_SCLIP, GLE_ASCLIP, GLE_SMARK, and GLE_AUTL.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:D      0 {
2:D      0 { Graphics Low End
3:D      0 {
4:D      0 { Module = GLE_RAS_OUT
5:D      0 { Programmer = BJS
6:D      0 { Date = 11-05-82
7:D      0 {
8:D      0 { Purpose: To provide device handler routines for raster devices.
9:D      0 {
10:D     0 { Rev history
11:D     0 { Created - 11-05-82
12:D     0 { Modified - 02-14-84 8DS (added code for gator black/white)
13:D     0 {
14:D     ( (c) Copyright Hewlett-Packard Company, 1983.
15:D     All rights are reserved. Copying or other
16:D     reproduction of this program except for archival
17:D     purposes is prohibited without the prior
18:D     written consent of Hewlett-Packard Company.
19:D
20:D
21:D     RESTRICTED RIGHTS LEGEND
22:D
23:D     Use, duplication, or disclosure by the Government
24:D     is subject to restrictions as set forth in
25:D     paragraph (b) (3) (B) of the Rights in Technical
26:D     Data and Computer Software clause in
27:D     DAR 7-104.9(a).
28:D
29:D     HEWLETT-PACKARD COMPANY
30:D     Fort Collins, Colorado
31:D
32:D     0 $search 'GLE_TYPES',
33:D     0 'GLE_STEXT',
34:D     0 'ASM_STEXT',
35:D     0 'GLE_SCLIP',
36:D     0 'ASM_SCLIP',
37:D     0 'GLE_SHARK',
38:D     0 'GLE_AUTL',
39:D     0 'RGL's
40:D
41:D     0 $modals
42:D     0 $include 'OPTIONS'$ ( ***** COMPILER OPTIONS ***** )
43:D
44:D     { This include file specifies range checking, debug and other compiler
45:D     options for the graphics library
46:D
47:D     0 $debug OFF$
48:D     0 $range OFF$
49:D     0 $copyright 'COPYRIGHT 1984 8Y HEWLETT-PACKARD COMPANY'S
50:D     0 $FIDAT_HDW TEST$
51:D
52:D
53:D     0 $include 'OPTIONS'$ ( ***** COMPILER OPTIONS ***** )
54:D     0 $inenum 8000$
55:D
56:D     0 module gle_ras_out;
57:D
58:D     1 import gle_types,sysglobals,sysdevs,gle_aras_out;
59:D
60:D

```

```

6006:D 1 export
6007:D
6008:D 1 procedure gle_init_raster_output ( gcb : graphics_control_block_ptr);
6009:D 1 procedure gator_clear ( gcb : graphics_control_block_ptr );
6010:D
6011:D 1 type
6012:D 1 raster_byte = 0..255;
6013:D 1 raster_code_space = packed array [1..240] of raster_byte;
6014:D 1 dither_type = packed array [0..15] of raster_byte;
6015:D 1 cmap_def = packed record
6016:D 1 map_red : gle_shortint;
6017:D 1 map_grn : gle_shortint;
6018:D 1 map_blu : gle_shortint;
6019:D 1 end;
6020:D
6021:D 1 system_cmap_def = packed array [0..15] of cmap_def;
6022:D
6023:D
6024:D 1 raster_device_rec_ptr = ^ raster_device_rec;
6025:D 1 raster_device_rec =
6026:D 1 record
6027:D 1 addr1 : anyptr ;
6028:D 1 addr2 : anyptr ;
6029:D 1 addr3 : anyptr ;
6030:D 1 n3 : gle_shortint ;
6031:D 1 devicetype : gle_shortint ;
6032:D 1 deviceaddress : integer ;
6033:D 1 monitortype : gle_shortint ;
6034:D 1 plane1_addr : anyptr ;
6035:D 1 plane1_offset : integer ;
6036:D 1 plane2_offset : integer ;
6037:D 1 plane3_offset : integer ;
6038:D 1 n_lines : gle_shortint ;
6039:D 1 gspacing : gle_shortint ;
6040:D 1 bytesperline : gle_shortint ;
6041:D 1 hard_xmax : gle_shortint ;
6042:D 1 hard_ymax : gle_shortint ;
6043:D 1 red_intensity : gle_shortint ;
6044:D 1 grn_intensity : gle_shortint ;
6045:D 1 blu_intensity : gle_shortint ;
6046:D 1 dither_pattern : dither_type;
6047:D 1 cursor_x : gle_shortint ;
6048:D 1 cursor_y : gle_shortint ;
6049:D 1 area_draw_mode : gle_shortint ;
6050:D 1 pen_draw_mode : gle_shortint ;
6051:D 1 linepattern : gle_shortint ;
6052:D 1 pen_number : gle_shortint ;
6053:D 1 cpen : gle_shortint ;
6054:D 1 oldpattern : gle_shortint ;
6055:D 1 rgltemp1 : integer ;
6056:D 1 rgltemp2 : integer ;
6057:D 1 rgltemp3 : integer ;
6058:D 1 rgltemp4 : integer ;
6059:D 1 rgltemp5 : integer ;
6060:D 1 repeatrate : gle_shortint ;
6061:D 1 repeatcount : gle_shortint ;
6062:D 1 index : integer ;
6063:D 1 softvec : raster_code_space ;
6064:D 1 system_cmap : system_cmap_def ;
6065:D 1 brightness_sequence : packed array [0..15] of gle_shortint;

```



```

8066:D      1      count      : packed array [0..15] of gle_shortint;
8067:D      1      cmap_address : integer;
8068:D      1      end;
8069:S
8070:D      1 const
8071:D      1 packed_pixel_odd_byte_display = 0;
8072:D      1 packed_pixel_display = 1;
8073:D      1 packed_pixel_3_plane_display = 2;
8074:D      1 byte_per_pixel_display = 3;
8075:D      1 gator_display = 4;
8076:S
8077:D      1 dominant = 3;
8078:D      1 erase = 0;
8079:D      1 non_dominant = 7;
8080:D      1 compliment = 10;
8081:D      1 clr_with_lm = 128;
8082:D      1 one_with_lm = 255;
8083:S
8084:D      1 implement
8085:S
8086:D      1 import gle_stext,
8087:D      1      gle_astext,
8088:D      1      gle_scilp,
8089:D      1      gle_ascilp,
8090:D      1      gle_smark,
8091:D      1      gle_aul1;
8092:S
8093:D      1 type
8094:D      1 word_array = packed array [1..maxint] of gle_shortint;
8095:D      1 fixed_word_array = array [0..13] of gle_shortint;
8096:D      1 map_array = array [1..maxint] of integer;
8097:D      1 wd_ptr = ^word_array;
8098:D      1 byte_array = packed array [0..maxint] of raster_byte;
8099:S
8100:D      1 var
8101:D      -4 1 fb_ptr : ^byte_array;
8102:D      -8 1 fb_ptr_ptr : ^integer;
8103:D      -8 1
8104:S
8105:S
8106:D      -8 1 (const)
8107:D      -8 1 {init_crt = fixed_word_array [22,16,21,01,48,11,48,48,0,15,0,0,0,0];{17"}}
8108:D      -8 1 {init_crt = fixed_word_array [20,16,17,2,48,11,48,48,0,15,0,0,0,0];{19"}}
8109:D      -8 1 {init_crt = fixed_word_array [20,16,16,4,48,11,48,48,0,15,0,0,32,0];{17"-NEW}}
8110:S
8111:D      4
8112:S
8113:D      1 PROCEDURE wait_ready;
8114:S
8115:C      2 BEGIN
8116:C      2 REPEAT
8117:C      3 UNTIL status^.notbusy
8118:C      3 END;
8119:S
8120:D      1 PROCEDURE set_rule ( rule : INTEGER );
8121:S
8122:C      2 BEGIN
8123:C      2 replregcopy := rule;
8124:C      2 wait_ready;
8125:C      2 rule_reg^ := rule;

```

```

8126:C      2 END;
8127:S
8128:D      1 (PROCEDURE set_wm ( rule : INTEGER );
8129:D      1
8130:S      BEGIN
8131:S      writeregcopy := rule;
8132:S      wait_ready;
8133:S      ww_reg^ := rule;
8134:D      -8 1 END;
8135:S
8136:D      1 PROCEDURE set_width ( width : INTEGER );
8137:S
8138:C      2 BEGIN
8139:C      2 windowregcopy := width;
8140:C      2 wait_ready;
8141:C      2 width_reg^ := width;
8142:C      2 END;
8143:S
8144:D      1 procedure gator_fill_index_color ( gcb : graphics_control_block_ptr );
8147:S
8148:D      2 type
8149:D      2 dpt1 = packed array [0..15] of boolean;
8150:D      2 dpt2 = packed array [0..16] of dpt1;
8151:D      2
8152:D      -4 2 var
8153:D      -8 2 i : integer;
8154:D      -8 2 j : integer;
8155:D      -12 2 r : integer;
8156:D      -16 2 g : integer;
8157:D      -20 2 b : integer;
8158:S
8159:D      -20 2 const
8160:D      -20 2 t = true;
8161:D      -20 2 f = false;
8162:D      -20 2 dp = dpt2[
8163:D      -20 2 dpt1[f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f],
8164:D      -20 2 dpt1[t,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f],
8165:D      -20 2 dpt1[t,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f],
8166:D      -20 2 dpt1[t,f,t,f,f,f,f,f,f,f,f,f,f,f,f,f],
8167:D      -20 2 dpt1[t,f,t,f,f,f,f,f,f,f,f,f,f,f,f,f],
8168:D      -20 2 dpt1[t,f,t,f,f,f,f,f,f,f,f,f,f,f,f,f],
8169:D      -20 2 dpt1[t,f,t,f,f,f,f,f,f,f,f,f,f,f,f,f],
8170:D      -20 2 dpt1[t,f,t,f,f,f,f,f,f,f,f,f,f,f,f,f],
8171:D      -20 2 dpt1[t,f,t,f,f,f,f,f,f,f,f,f,f,f,f,f],
8172:D      -20 2 dpt1[t,f,t,f,f,f,f,f,f,f,f,f,f,f,f,f],
8173:D      -20 2 dpt1[t,f,t,f,f,f,f,f,f,f,f,f,f,f,f,f],
8174:D      -20 2 dpt1[t,f,t,f,f,f,f,f,f,f,f,f,f,f,f,f],
8175:D      -20 2 dpt1[t,t,t,f,f,f,f,f,f,f,f,f,f,f,f,f],
8176:D      -20 2 dpt1[t,t,t,f,f,f,f,f,f,f,f,f,f,f,f,f],
8177:D      -20 2 dpt1[t,t,t,f,f,f,f,f,f,f,f,f,f,f,f,f],
8178:D      -20 2 dpt1[t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t]];
8179:S
8180:C      2 begin
8181:C      2 with gcb^, raster_device_rec_ptr(dev_dev_stuff)^ do
8182:C      3 begin
8183:C      3 if info1 = 0 then (set up dither pattern)
8184:C      4 begin
8185:C      4 r := info2;

```



```

8186:C      4      g := info3;
8187:C      4      b := info4;
8188:S
8189:C      4      current_polygon_red := r;
8190:C      4      current_polygon_green := g;
8191:C      4      current_polygon_blue := b;
8192:S
8193:C      4      r := (r+32) div 64;
8194:C      4      g := (g+32) div 64;
8195:C      4      b := (b+32) div 64;
8196:S
8197:C      4      red_intensity := r;
8198:C      4      grn_intensity := g;
8199:C      4      blu_intensity := b;
8200:S
8201:C      4      i:=r;
8202:C      4      if i<g then i:=g;
8203:C      4      if i<b then i:=b;
8204:S
8205:C      4      for j:=0 to 15 do dither_pattern[j]:=0;
8206:C      4      for j:=0 to 15 do if dp[i,j] then dither_pattern[j]:=1;
8207:S
8208:C      4      store_dit(gcb);
8209:C      4      end
8210:C      4      else
8211:C      4      if info1 = 1 then
8212:C      5      begin
8213:C      5      for i := 0 to 15 do
8214:C      6      dither_pattern[i] := info2;
8215:C      5      store_dit(gcb);
8216:C      5      end;
8217:S
8218:C      3      end;
8219:C      2 end;
8220:S
8221:S
8222:S
8223:D      1 procedure gator_clear ( gcb : graphics_control_block_ptr );
8224:S
8225:D      -4 2 var i : integer;
8226:S
8227:C      2 begin
8228:C      2 if gcb^.info2 = 0 then set_rule(clr_with_LM);
8229:C      3 else set_rule(one_with_LM);
8230:C      2 set_width(-1024);
8231:C      2 with gcb^.raster_device_rec_ptr(dev_dep_stuff)^ do
8232:C      3 begin
8233:C      4 for i := 0 to hard_ymax do (fix to allow for changing ymax)
8234:C      4 begin
8235:C      4 wait_ready;
8236:C      4 fb_ptr^[i*1024] := 0;
8237:C      4 end;
8238:C      3 end;
8239:C      2 set_rule(dominant);
8240:C      2 end;
8241:S
8242:S
8243:S
8244:D      1 procedure setup_gator(gcb : graphics_control_block_ptr);
8245:S

```

```

8246:D      2 var
8247:D      -96 2 color_list : packed array [0..47] of gle_shortint;
8248:D      -100 2 i : integer;
8249:S
8250:C      2 begin
8251:C      2 (with gcb^ do
8252:C      2 (if info1 = mgator_c then begin
8253:C      2 set_wm(0);
8254:C      2 ( I moved this as a fix )
8255:S      (blink1_reg := 255;
8256:S      blink2_reg := 255;
8257:S      end;))
8258:C      2 set_6845s (gcb);
8259:C      2 (for i := 0 to 13 do
8260:C      2 begin
8261:C      2 crt^[i] := i;
8262:C      2 crt^[2] := init_crt[i]; SETUP 6845
8263:C      2 end;))
8264:C      2 status.bit0 := write_to_status;
8265:C      2 (color_list[0] := 0;
8266:C      2 color_list[1] := 0;
8267:C      2 color_list[2] := 0;
8268:C      2 i := 0;
8269:C      2 repeat
8270:C      2 i := i + 3;
8271:C      2 color_list[i] := 1023;
8272:C      2 color_list[i+1] := 0;
8273:C      2 color_list[i+2] := 0;
8274:C      2 until i = 45;))
8275:C      2 with gcb^ do
8276:C      3 begin
8277:C      3 info1 := 0;
8278:C      3 info2 := 1;
8279:C      3 info_ptr1 := addr(color_list);
8280:C      3 call (define_color_map,gcb);
8281:C      3 end;
8282:C      2 set_rule(dominant);
8283:C      2 end;
8284:S
8285:S      (procedure rdummy ( anyvar a,b : anyptr);
8286:S
8287:S      begin
8288:D      -8 1 end;))
8289:S
8290:D      1 procedure rdummy_proc ( gcb : graphics_control_block_ptr );
8291:S
8292:D      2 VAR
8293:D      -2 2 I GLE_SHORTINT;
8294:S
8295:C      2 begin
8296:C      2 with gcb^.raster_device_rec_ptr(dev_dep_stuff)^ do
8297:C      2 (for i := 0 to 15 do
8298:C      3 Writeln(Listing,'DIT['',i:0,'] = ',DITHER_PATTERN[i]);)
8299:C      2 end;
8300:S
8301:D      1 procedure rget_polygon_info ( gcb : graphics_control_block_ptr );
8302:S
8303:C      2 begin ( only solid fill supported )
8304:C      2 with gcb^ do
8305:C      3 if info2 = polygon_solid_fill then error_return := 0

```



```

8306:C      4      else                      error_return := 1;
8307:C      2      end;
8308:S
8309:D      1      procedure rdefine_color_map ( gcb : graphics_control_block_ptr );
8310:S
8311:D      2      type
8312:D          color_map_def = packed array [0..15] of gle_shortint;
8313:D          color_map_ptr = ^color_map_def;
8314:D          color_data_def = packed array [0..47] of gle_shortint;
8315:D          color_data_ptr_def = ^color_data_def;
8316:S
8317:D      2      var
8318:D          color_map : color_map_ptr;
8319:D          temp_color_map : color_map_def;
8320:D          n,i : gle_shortint;
8321:D          temp : integer;
8322:D          brt : array [0..15] of integer;
8323:D          color_data_ptr : color_data_ptr_def;
8324:S
8325:C      2      begin
8326:C          with gcb^, raster_device_rec_ptr(dev_dep_stuff)^ do
8327:C              begin
8328:C                  color_data_ptr := info_ptr1;
8329:C                  for i := info1 to info2 do
8330:C                      with system_cmap[i] do
8331:C                          begin
8332:C                              n := (i - info1)*3;
8333:C                              map_red := color_data_ptr^[n];
8334:C                              map_grn := color_data_ptr^[n+1];
8335:C                              map_blu := color_data_ptr^[n+2];
8336:C                              temp_color_map[i] := (15-map_red div 64)*256 +
8337:C                                                      (15-map_grn div 64) * 16 +
8338:C                                                      (15-map_blu div 64);
8339:C                          end;
8340:C                      end;
8341:C          3      if color_map_support = yes then
8342:C              begin
8343:C                  color_map := anyptr(cmap_address);
8344:C                  rawait_blanking(gcb);
8345:C                  for i := info1 to info2 do
8346:C                      color_map^[i] := temp_color_map[i];
8347:C                  end;
8348:S
8349:C          3      for i := 0 to 15 do
8350:C              with system_cmap[i] do
8351:C                  begin
8352:C                      brt[i] := map_red*23+map_grn*69+map_blu*8;
8353:C                      brightness_sequence[i] := 1;
8354:C                  end;
8355:C          3      for n := 0 to 14 do
8356:C              for i := n+1 to 15 do
8357:C                  if brt[i]>brt[n] then
8358:C                      begin
8359:C                          temp := brt[i];
8360:C                          brt[i] := brt[n];
8361:C                          brt[n] := temp;
8362:C                          temp := brightness_sequence[i];
8363:C                          brightness_sequence[i] := brightness_sequence[n];
8364:C                          brightness_sequence[n] := temp;
8365:C                      end;

```

```

8366:S          (FOR N := 0 TO 15 DO
8367:S              WRITELN(LISTING,'N = ',N,'8RI[N]= ',8RIGHTNESS_SEQUENCE[N]);
8368:S          FOR N := 0 TO 15 DO
8369:S              WITH SYSTEM_CMAP[N] DO
8370:S                  3      WRITELN(LISTING,'N = ',N,'CMAP[N]= ',MAP_RED:8,MAP_GRN:8,MAP_BLU:8);
8371:S          3      end;
8372:C      2      end;
8373:S
8374:D      1      procedure gle_init_raster_output ( gcb : graphics_control_block_ptr);
8375:S
8376:D      2      var
8377:D          cnt : gle_shortint;
8378:D          i,id : integer;
8379:S
8380:C      2      begin
8381:C          with gcb^, raster_device_rec_ptr(dev_dep_stuff)^ do
8382:C              begin
8383:C                  if (info1 = m9837a) then                      (or (info1 = mgator_c))
8384:C                      begin
8385:C                          fb_ptr_ptr := info_ptr1;
8386:C                          fb_ptr := anyptr(fb_ptr_ptr^);
8387:C                          rule_reg := anyptr(info2 + (hex('4008')));
8388:C                          width_reg := anyptr(info2 + (hex('400c')));
8389:C                          status := anyptr(info2 + (hex('4000')));
8390:C                          crt := anyptr(info2 + (hex('6000')));
8391:C                          (if info1 = mgator_c then ww_reg := anyptr(info2 + (hex('4108'))));
8392:C                      end;
8393:C          3      error_return := 0;
8394:C          3      rgcbinit (-gcb);
8395:S
8396:C          3      move                := rmove;
8397:C          3      draw                 := rdraw;
8398:C          3      clear                := rclear;
8399:C          3      text                 := gle_soft_text;
8400:C          3      clip_limits          := gle_soft_clip_limits;
8401:C          3      char_size            := gle_soft_char_size;
8402:C          3      text_spacing        := gle_soft_text_spacing;
8403:C          3      text_dir             := gle_soft_text_dir;
8404:C          3      text_just            := gle_soft_text_just;
8405:C          3      marker              := gle_soft_marker;
8406:C          3      marker_size         := gle_soft_marker_size;
8407:C          3      set_marker           := gle_soft_set_marker;
8408:C          3      index_color         := rset_color;
8409:C          3      linestyle           := rlinestyle;
8410:C          3      await_blanking       := rawait_blanking;
8411:C          3      linewidth           := rdummy_proc;
8412:C          3      inq_plp2             := rget_plp2;
8413:C          3      get_polygon_info     := rget_polygon_info;
8414:C          3      calc_soft_text_xform := gle_text_xform;
8415:C          3      fill_index_color     := rfill_index_color;
8416:C          3      graphics_on_off     := rgraphics_on_off;
8417:C          3      cursor              := rcursor;
8418:C          3      define_drawing_mode := rdefine_drawing_mode;
8419:C          3      polygon             := rpolygon;
8420:C          3      define_color_map     := rdefine_color_map;
8421:C          3      buffer_mode         := rdummy_proc;
8422:C          3      output_escapei      := rdummy_proc;
8423:C          3      output_escapeo      := rdummy_proc;
8424:C          3      flush_buffer        := rdummy_proc;
8425:C          3      get_raster          := rget_raster;

```



```

8426:S      3      soft_font_ptr      := addr(font);
8427:C
8428:S
8429:C      3      case info1 of
8430:C          munknown : { reserved } ;
8431:C          m9816a : { 9816 }
8432:C          begin
8433:C              display_name := '9816A';
8434:C              display_name_char_count := 5;
8435:C              display_res_x := 2.375;          ( 168mm X 126mm )
8436:C              display_res_y := 2.37301587301587;
8437:C          end;
8438:C          m9826a : { 9826 }
8439:C          begin
8440:C              display_name := '9826A';
8441:C              display_name_char_count := 5;
8442:C              display_res_x := 3.325;          ( 120mm X 90mm )
8443:C              display_res_y := 3.32222222222222;
8444:C          end;
8445:C          m9836a : { 9836 }
8446:C          begin
8447:C              display_name := '9836A';
8448:C              display_name_char_count := 5;
8449:C              display_res_x := 2.43333333333333; ( 210mm X 160mm )
8450:C              display_res_y := 2.43125;
8451:C          end;
8452:C          m9836c : { 9836C }
8453:C          begin
8454:C              display_name := '9836C';
8455:C              display_name_char_count := 5;
8456:C              display_res_x := 2.35483870967742; ( 217mm X 163mm )
8457:C              display_res_y := 2.38650306748466;
8458:C          end;
8459:C          m98627a : { 98627 }
8460:C          begin
8461:C              display_name := '98627A';
8462:C              display_name_char_count := 6;
8463:C              deviceaddress := info2;
8464:C              display_res_x := 3.33333333333333;
8465:C              display_res_y := 3.33333333333333;
8466:C          end;
8467:C          m9837a : { Gator 81ack/White }
8468:C          begin
8469:C              display_name := '9837a';
8470:C              display_name_char_count := 5;
8471:C              display_res_x := 3.29166666666667; ( 312mm X 234mm )
8472:C              display_res_y := 3.28205128205128;
8473:C              if info3 = I then (added to detect current)
8474:C                  begin (ymax.)
8475:C                      n_lines := 768;
8476:C                      hard_ymax := 767;
8477:C                  end;
8478:C              await_blanking := rdummy_proc;
8479:C              setup_gator(gcb);
8480:C              clear := gator_clear;
8481:C              fill_index_color := gator_fill_index_color;
8482:C              define_color_map := rdummy_proc;
8483:C          end;
8484:S      (mgator_c : Gator Color
8485:S      begin

```

```

8486:S      display_name := 'Gator';
8487:S      display_name_char_count := 5;
8488:S      display_res_x := 3.29166666666667;          312mm X 234mm
8489:S      display_res_y := 3.28205128205128;
8490:S      await_blanking := rdummy_proc;
8491:S      setup_gator(gcb);
8492:S      clear := gator_clear;
8493:S      define_color_map := rdummy_proc;
8494:S      INFO2 := 0;
8495:S      GATOR_CLEAR(GCB);          WHY IS THIS HERE ??
8496:C      end;
8497:C      end; {of case }
8498:S
8499:S
8500:S      3      display_handler_name := 'RASTER';
8501:S      3      display_handler_char_count := 6;
8502:S
8503:C      3      display_min_x := 0;
8504:C      3      display_min_y := 0;
8505:C      3      display_max_x := hard_xmax;
8506:C      3      display_max_y := hard_ymax;
8507:S
8508:S
8509:C      3      info1 := 0; info2 := display_max_x;
8510:C      3      info3 := 0; info4 := display_max_y;
8511:C      3      gle_soft_clip_limits ( gcb ); { set default clipping limits }
8512:S
8513:C      3      pen_draw_mode := 0;
8514:C      3      area_draw_mode := 0;
8515:C      3      pen_number := 1;
8516:C      3      info1 := 1;
8517:C      3      rset_color(gcb);
8518:C      3      end;
8519:C      2      end;
8520:S
8521:C      1      end. { output }
8522:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

GLE_SCLIP

Description

GLE_SCLIP provides a routine which sets GLE clip limits.

Requirements

GLE_TYPES.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:D      0 { Graphics Low End }
2:D      0 {
3:D      0 { Module = GLE_SCLIP }
4:D      0 { Programmer = BJS }
5:D      0 { Date = 10-10-82 }
6:D      0 {
7:D      0 { Purpose: To provide software clipping routines. }
8:D      0 {
9:S      0 { Rev history }
10:D     0 { Created = 10-10-82 }
11:D     0 { Modified = XX-XX-XX }
12:D     0 {
13:S     (
14:S     (c) Copyright Hewlett-Packard Company, 1983.
15:S     All rights are reserved. Copying or other
16:S     reproduction of this program except for archival
17:S     purposes is prohibited without the prior
18:S     written consent of Hewlett-Packard Company.
19:S
20:S
21:S     RESTRICTED RIGHTS LEGEND
22:S
23:S     Use, duplication, or disclosure by the Government
24:S     is subject to restrictions as set forth in
25:S     paragraph (b) (3) (B) of the Rights in Technical
26:S     Data and Computer Software clause in
27:S     DAR 7-104.9(a).
28:S
29:S     HEWLETT-PACKARD COMPANY
30:D     Fort Collins, Colorado }
31:S
32:S
33:D     0 $modcls$
34:D     0 $search 'GLE TYPES'$
35:D     0 $include 'OPTIONS'$ ( ***** COMPILER OPTIONS ***** )
36:S     { This include file specifies range checking, debug and other compiler
37:D     0 options for the graphics library }
38:S
39:D     0 $debug OFF$
40:D     0 $range OFF$
41:D     0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
42:D     0 $FLOAT_ROW TEST$
43:S
44:S
45:S
46:S
47:D     0 $include 'OPTIONS'$ ( ***** COMPILER OPTIONS ***** )
48:D     0 $linenum 6000$
49:D     0 module gle_sclip;
50:D     1 import gle_types;
51:D     1 export
52:D     1 procedure gle_soft_clip_limits ( gcb : graphics_control_block_ptr);
53:D     1 implement
54:D
55:D
56:D
57:D
58:D
59:D
60:D
61:D
62:D
63:D
64:D
65:D
66:D
67:D
68:D
69:D
70:D
71:D
72:D
73:D
74:D
75:D
76:D
77:D
78:D
79:D
80:D
81:D
82:D
83:D
84:D
85:D
86:D
87:D
88:D
89:D
90:D
91:D
92:D
93:D
94:D
95:D
96:D
97:D
98:D
99:D
100:D
101:D
102:D
103:D
104:D
105:D
106:D
107:D
108:D
109:D
110:D
111:D
112:D
113:D
114:D
115:D
116:D
117:D
118:D
119:D
120:D
121:D
122:D
123:D
124:D
125:D
126:D
127:D
128:D
129:D
130:D
131:D
132:D
133:D
134:D
135:D
136:D
137:D
138:D
139:D
140:D
141:D
142:D
143:D
144:D
145:D
146:D
147:D
148:D
149:D
150:D
151:D
152:D
153:D
154:D
155:D
156:D
157:D
158:D
159:D
160:D
161:D
162:D
163:D
164:D
165:D
166:D
167:D
168:D
169:D
170:D
171:D
172:D
173:D
174:D
175:D
176:D
177:D
178:D
179:D
180:D
181:D
182:D
183:D
184:D
185:D
186:D
187:D
188:D
189:D
190:D
191:D
192:D
193:D
194:D
195:D
196:D
197:D
198:D
199:D
200:D
201:D
202:D
203:D
204:D
205:D
206:D
207:D
208:D
209:D
210:D
211:D
212:D
213:D
214:D
215:D
216:D
217:D
218:D
219:D
220:D
221:D
222:D
223:D
224:D
225:D
226:D
227:D
228:D
229:D
230:D
231:D
232:D
233:D
234:D
235:D
236:D
237:D
238:D
239:D
240:D
241:D
242:D
243:D
244:D
245:D
246:D
247:D
248:D
249:D
250:D
251:D
252:D
253:D
254:D
255:D
256:D
257:D
258:D
259:D
260:D
261:D
262:D
263:D
264:D
265:D
266:D
267:D
268:D
269:D
270:D
271:D
272:D
273:D
274:D
275:D
276:D
277:D
278:D
279:D
280:D
281:D
282:D
283:D
284:D
285:D
286:D
287:D
288:D
289:D
290:D
291:D
292:D
293:D
294:D
295:D
296:D
297:D
298:D
299:D
300:D
301:D
302:D
303:D
304:D
305:D
306:D
307:D
308:D
309:D
310:D
311:D
312:D
313:D
314:D
315:D
316:D
317:D
318:D
319:D
320:D
321:D
322:D
323:D
324:D
325:D
326:D
327:D
328:D
329:D
330:D
331:D
332:D
333:D
334:D
335:D
336:D
337:D
338:D
339:D
340:D
341:D
342:D
343:D
344:D
345:D
346:D
347:D
348:D
349:D
350:D
351:D
352:D
353:D
354:D
355:D
356:D
357:D
358:D
359:D
360:D
361:D
362:D
363:D
364:D
365:D
366:D
367:D
368:D
369:D
370:D
371:D
372:D
373:D
374:D
375:D
376:D
377:D
378:D
379:D
380:D
381:D
382:D
383:D
384:D
385:D
386:D
387:D
388:D
389:D
390:D
391:D
392:D
393:D
394:D
395:D
396:D
397:D
398:D
399:D
400:D
401:D
402:D
403:D
404:D
405:D
406:D
407:D
408:D
409:D
410:D
411:D
412:D
413:D
414:D
415:D
416:D
417:D
418:D
419:D
420:D
421:D
422:D
423:D
424:D
425:D
426:D
427:D
428:D
429:D
430:D
431:D
432:D
433:D
434:D
435:D
436:D
437:D
438:D
439:D
440:D
441:D
442:D
443:D
444:D
445:D
446:D
447:D
448:D
449:D
450:D
451:D
452:D
453:D
454:D
455:D
456:D
457:D
458:D
459:D
460:D
461:D
462:D
463:D
464:D
465:D
466:D
467:D
468:D
469:D
470:D
471:D
472:D
473:D
474:D
475:D
476:D
477:D
478:D
479:D
480:D
481:D
482:D
483:D
484:D
485:D
486:D
487:D
488:D
489:D
490:D
491:D
492:D
493:D
494:D
495:D
496:D
497:D
498:D
499:D
500:D
501:D
502:D
503:D
504:D
505:D
506:D
507:D
508:D
509:D
510:D
511:D
512:D
513:D
514:D
515:D
516:D
517:D
518:D
519:D
520:D
521:D
522:D
523:D
524:D
525:D
526:D
527:D
528:D
529:D
530:D
531:D
532:D
533:D
534:D
535:D
536:D
537:D
538:D
539:D
540:D
541:D
542:D
543:D
544:D
545:D
546:D
547:D
548:D
549:D
550:D
551:D
552:D
553:D
554:D
555:D
556:D
557:D
558:D
559:D
560:D
561:D
562:D
563:D
564:D
565:D
566:D
567:D
568:D
569:D
570:D
571:D
572:D
573:D
574:D
575:D
576:D
577:D
578:D
579:D
580:D
581:D
582:D
583:D
584:D
585:D
586:D
587:D
588:D
589:D
590:D
591:D
592:D
593:D
594:D
595:D
596:D
597:D
598:D
599:D
600:D
601:D
602:D
603:D
604:D
605:D
606:D
607:D
608:D
609:D
610:D
611:D

```

```

6012:D    1 procedure gle_soft_clip_limits ( gcb : graphics_control_block_ptr );
6013:S
6014:C    2 begin
6015:C    2   with gcb^ do
6016:C    3   begin
6017:C    3     clip_limits_xmin := info1;
6018:C    3     clip_limits_xmax := info2;
6019:C    3     clip_limits_ymin := info3;
6020:C    3     clip_limits_ymax := info4;
6021:C    3   end;
6022:C    2 end;
6023:S
6024:C    1 end ( module gle_sclip )
6025:S

```

No errors. No warnings.
 ***** Nonstandard language features enabled *****

GLE_SMARK

Description

GLE_SMARK provides software marker routines.

Requirements

GLE_TYPES, and GLE_AUTL.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```
1:S
2:D 0 {
3:D 0 { Graphics Low End }
4:D 0 {
5:D 0 { Module = GLE_SMARK
6:D 0 { Programmer = BJ5
7:D 0 { Date = 10-15-82
8:D 0 {
9:D 0 { Purpose: To provide software marker routines.
10:S }
11:D 0 { Rev history
12:D 0 { Created - 10-15-82 BJ5
13:D 0 { Modified - XX-XX-XX
14:S }
15:S {
16:S { (c) Copyright Hewlett-Packard Company, 1983.
17:S { All rights are reserved. Copying or other
18:S { reproduction of this program except for archival
19:S { purposes is prohibited without the prior
20:S { written consent of Hewlett-Packard Company.
21:S }
22:S
23:S
24:S
25:S
26:S
27:S
28:S
29:S
30:S
31:D 0
32:S
33:S
34:D 0 $modcal$
35:D 0 $search 'GLE_AUTL'
36:D 0 $GLE_TYPES$
37:S
38:S
39:D 0 $include 'OPTIONS'$ { ***** COMPILER OPTIONS ***** }
40:S { This include file specifies range checking, debug and other compiler
41:S { options for the graphics library }
42:S
43:D 0 $debug OFF$
44:D 0 $range OFF$
45:D 0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
46:D 0 $FLOAT_HDW TEST$
47:S
48:S
49:D 0 $include 'OPTIONS'$ { ***** COMPILER OPTIONS ***** }
5000:D 0 $linenum 5000$
5001:S
5002:S
5003:S
5004:D 0 module gle_smarm;
5005:S
5006:D 1 import gle_types;
5007:S
5008:D 1 export
5009:D 1 procedure gle_soft_marker ( gcb : graphics_control_block_ptr);
5009:D 1 procedure gle_soft_set_marker ( gcb : graphics_control_block_ptr);
```

```
5010:D 1 procedure gle_soft_marker_size ( gcb : graphics_control_block_ptr);
5011:S
5012:S
5013:S
5014:D 1 import gle_autl;
5015:S
5016:D 1 procedure gle_soft_marker ( gcb : graphics_control_block_ptr);
5017:S
5018:D 2 const
5019:D 2 marker_cell_shift = -2; { part of marker viewing transformation }
5020:S
5021:D 2 type marker_data = -2..2;
5022:D 2 index_def = 0..50;
5023:D 2 marker_record = packed record
5024:D 2 x_pos : marker_data;
5025:D 2 y_pos : marker_data;
5026:D 2 penup : boolean;
5027:D 2 end;
5028:D 2 mark_def = packed array [1..47] of marker_record;
5029:D 2 marker_index_def = packed array [1..10] of index_def;
5030:S
5031:D 2 const
5032:D 2 markers =
5033:D 2 mark_def [
5034:D 2 marker_record [x_pos : 0, y_pos : 0, penup : true], {1}
5035:D 2 marker_record [x_pos : 0, y_pos : 0, penup : false], {2}
5036:D 2 marker_record [x_pos : 0, y_pos : 2, penup : true], {3}
5037:D 2 marker_record [x_pos : 0, y_pos : -2, penup : false], {4}
5038:D 2 marker_record [x_pos : 2, y_pos : 0, penup : true], {5}
5039:D 2 marker_record [x_pos : -2, y_pos : 0, penup : false], {6}
5040:D 2 marker_record [x_pos : 2, y_pos : 0, penup : true], {7}
5041:D 2 marker_record [x_pos : -2, y_pos : 0, penup : false], {8}
5042:D 2 marker_record [x_pos : 2, y_pos : 2, penup : true], {9}
5043:D 2 marker_record [x_pos : -2, y_pos : -2, penup : false], {10}
5044:D 2 marker_record [x_pos : 2, y_pos : 2, penup : true], {11}
5045:D 2 marker_record [x_pos : -2, y_pos : -2, penup : false], {12}
5046:D 2 marker_record [x_pos : -1, y_pos : -2, penup : true], {13}
5047:D 2 marker_record [x_pos : 1, y_pos : -2, penup : false], {14}
5048:D 2 marker_record [x_pos : 2, y_pos : -1, penup : false], {15}
5049:D 2 marker_record [x_pos : -2, y_pos : 1, penup : true], {16}
5050:D 2 marker_record [x_pos : 2, y_pos : 2, penup : false], {17}
5051:D 2 marker_record [x_pos : -2, y_pos : 1, penup : false], {18}
5052:D 2 marker_record [x_pos : 2, y_pos : -1, penup : false], {19}
5053:D 2 marker_record [x_pos : -1, y_pos : -2, penup : false], {20}
5054:D 2 marker_record [x_pos : -2, y_pos : -2, penup : true], {21}
5055:D 2 marker_record [x_pos : 2, y_pos : 2, penup : false], {22}
5056:D 2 marker_record [x_pos : -2, y_pos : 2, penup : true], {23}
5057:D 2 marker_record [x_pos : 2, y_pos : -2, penup : false], {24}
5058:D 2 marker_record [x_pos : -2, y_pos : -2, penup : true], {25}
5059:D 2 marker_record [x_pos : 0, y_pos : 2, penup : true], {26}
5060:D 2 marker_record [x_pos : 2, y_pos : -2, penup : false], {27}
5061:D 2 marker_record [x_pos : -2, y_pos : -2, penup : false], {28}
5062:D 2 marker_record [x_pos : 0, y_pos : 2, penup : false], {29}
5063:D 2 marker_record [x_pos : -2, y_pos : 2, penup : true], {30}
5064:D 2 marker_record [x_pos : 2, y_pos : 2, penup : false], {31}
5065:D 2 marker_record [x_pos : -2, y_pos : 2, penup : false], {32}
5066:D 2 marker_record [x_pos : 2, y_pos : -2, penup : false], {33}
5067:D 2 marker_record [x_pos : 0, y_pos : 2, penup : true], {34}
5068:D 2 marker_record [x_pos : 2, y_pos : 0, penup : false], {35}
5069:D 2 marker_record [x_pos : 0, y_pos : -2, penup : false], {36}
```



```

5070:D      2      marker_record [x_pos :-2, y_pos : 0, penup : false], (38)
5071:D      2      marker_record [x_pos : 0, y_pos : 2, penup : false], (39)
5072:D      2      marker_record [x_pos :-2, y_pos : 2, penup : true], (40)
5073:D      2      marker_record [x_pos : 2, y_pos : 2, penup : false], (41)
5074:D      2      marker_record [x_pos :-2, y_pos : -2, penup : false], (42)
5075:D      2      marker_record [x_pos : 2, y_pos : -2, penup : false], (43)
5076:D      2      marker_record [x_pos :-2, y_pos : -2, penup : false], (44)
5077:D      2      marker_record [x_pos : 2, y_pos : 2, penup : false], (45)
5078:D      2      marker_record [x_pos : 2, y_pos : -2, penup : true], (46)
5079:D      2      marker_record [x_pos :-2, y_pos : 2, penup : false];
5080:S
5081:D      2      marker_index =
5082:D      2      marker_index_def [ (1) 1,
5083:D      2      (2) 3,
5084:D      2      (3) 7,
5085:D      2      (4) 13,
5086:D      2      (5) 22,
5087:D      2      (6) 26,
5088:D      2      (7) 30,
5089:D      2      (8) 35,
5090:D      2      (9) 40,
5091:D      2      (last) 48];
5092:S
5093:D      2      var
5094:D      2      1 : integer;
5095:D      2      s : packed array [1..1] of char;
5096:D      2      cpx : gle_shortint;
5097:D      2      cpy : gle_shortint;
5098:D      2      dx : gle_shortint;
5099:D      2      dy : gle_shortint;
5100:D      2      local_x_pos : gle_shortint;
5101:D      2      local_y_pos : gle_shortint;
5102:D      2      saved_char_width : integer;
5103:D      2      saved_char_height : integer;
5104:D      2      saved_text_cos_dir : integer;
5105:D      2      saved_text_sin_dir : integer;
5106:D      2      saved_line_pattern : integer;
5107:D      2      saved_linestyle : integer;
5108:D      2      saved_length : integer;
5109:D      2      saved_mode : integer;
5110:S
5111:C      2      begin
5112:C      2      with gcb^ do
5113:C      2      begin
5114:C      2      saved_mode := current_linestyle_mode;
5115:C      2      saved_length := current_pattern_length;
5116:C      2      saved_linestyle := current_linestyle;
5117:C      2      saved_line_pattern := current_linestyle_pattern;
5118:C      2      info1 := 0; info2 := 1;
5119:C      2      info3 := 0; info4 := 1;
5120:C      2      call ( linestyle, gcb );
5121:C      2      cpx := current_pos_x;
5122:C      2      cpy := current_pos_y;
5123:C      2      if marker_type > 9 then
5124:C      2      begin
5125:C      2      saved_char_width := char_width;
5126:C      2      saved_char_height := char_height;
5127:C      2      saved_text_cos_dir := text_cos_dir;
5128:C      2      saved_text_sin_dir := text_sin_dir;
5129:C      2      info1 := marker_width * 8;

```

```

5130:C      4      info2 := marker_height * 8;
5131:C      4      call ( char_size, gcb );
5132:C      4      info1 := 32768;
5133:C      4      info2 := 0;
5134:C      4      call ( text_dir, gcb );
5135:C      4      s[1] := chr(38*marker_type);
5136:C      4      end_x := cpx - (marker_width div 2);
5137:C      4      end_y := cpy - (marker_height div 2);
5138:C      4      call ( move, gcb );
5139:C      4      info_ptr1 := addr(s);
5140:C      4      info1 := 1;
5141:C      4      call ( text, gcb );
5142:S
5143:C      4      { restore char state }
5144:S
5145:C      4      info1 := saved_char_width;
5146:C      4      info2 := saved_char_height;
5147:C      4      call ( char_size, gcb );
5148:C      4      info1 := saved_text_cos_dir;
5149:C      4      info2 := saved_text_sin_dir;
5150:C      4      call ( text_dir, gcb );
5151:C      4      end
5152:C      4      else
5153:C      4      for i := marker_index[marker_type] to marker_index[marker_type+1]-1 do
5154:C      5      with markers[i] do
5155:C      6      begin
5156:C      6      local_x_pos := x_pos; { move packed field to temp to force type }
5157:C      6      local_y_pos := y_pos; { to gle_shortint }
5158:C      6      dx := gle_ishift((local_x_pos * marker_width), marker_cell_shift) + cpx;
5159:C      6      dy := gle_ishift((local_y_pos * marker_height), marker_cell_shift) + cpy;
5160:C      6      end_x := dx;
5161:C      6      end_y := dy;
5162:C      6      if penup then
5163:C      7      call ( move, gcb );
5164:C      7      else
5165:C      7      call ( draw, gcb );
5166:C      6      end;
5167:C      3      info1 := saved_linestyle;
5168:C      3      info2 := saved_length;
5169:C      3      info3 := saved_mode;
5170:C      3      info4 := saved_line_pattern;
5171:C      3      call ( linestyle, gcb ); { restore cp }
5172:C      3      end_x := cpx;
5173:C      3      end_y := cpy;
5174:C      3      call ( move, gcb );
5175:C      3      end;
5176:C      2      end;
5177:S
5178:C      1      procedure gle_soft_set_marker ( gcb : graphics_control_block_ptr );
5179:S
5180:C      2      begin
5181:C      2      with gcb^ do
5182:C      3      marker_type := info1;
5183:C      2      end;
5184:S
5185:C      1      procedure gle_soft_marker_size ( gcb : graphics_control_block_ptr );
5186:S
5187:C      2      begin
5188:C      2      with gcb^ do
5189:C      3      begin

```



```
S190:C      3      marker_width := info1;
S191:C      3      marker_height := info2;
S192:C      3      end;
S193:C      2 end;
S194:S
S195:C      1 end. ( module gle_smark )
S196:S
```

No errors. No warnings.

*****Nonstandard language features enabled*****

GLE_STEXT

Description

GLE_STEXT provides software text setup routines and the text size and rotation transformation.

Requirements

GLE_TYPES.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:D 0 {
2:D 0 { Graphics Low End
3:D 0 {
4:D 0 { Module = GLE_STEXT
5:D 0 { Programmer = BJS
6:D 0 { Date = 10-10-82
7:D 0 {
8:D 0 { Purpose: To provide software text routines.
9:S }
10:D 0 { Rev history
11:D 0 { Created - 10-10-82
12:D 0 { Modified - 6-28-83 BJS Removed soft_text from export text. This
13:D 0 { procedure is imported from gle_astext.
14:D 0 { 12-08-83 BDS Put Range Check on around calculation of
15:D 0 { dx and dy to eliminate random vectors.
16:S }
17:S
18:S { (c) Copyright Hewlett-Packard Company, 1983.
19:S All rights are reserved. Copying or other
20:S reproduction of this program except for archival
21:S purposes is prohibited without the prior
22:S written consent of Hewlett-Packard Company.
23:S
24:S
25:S RESTRICTED RIGHTS LEGEND
26:S
27:S Use, duplication, or disclosure by the Government
28:S is subject to restrictions as set forth in
29:S paragraph (b) (3) (B) of the Rights in Technical
30:S Data and Computer Software clause in
31:S DAR 7-104.9(a).
32:S
33:S HEWLETT-PACKARD COMPANY
34:D 0 Fort Collins, Colorado
35:S }
36:D 0 $modcal$
37:D 0 $search 'GLE_TYPES'$
37:D 0 $include 'OPTIONS'$ ( ***** COMPILER OPTIONS ***** )
38:S { This include file specifies range checking, debug and other compiler
39:S options for the graphics library
40:D 0
41:S
42:D 0 $debug OFF$
43:D 0 $range OFF$
44:D 0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
45:D 0 $FLOAT_HOW TEST$
46:S
47:S
48:S
49:S
50:D 0 $include 'OPTIONS'$ ( ***** COMPILER OPTIONS ***** )
4000:D 0 $linenum 4000$
4001:S
4002:D 0 module gle_stext.
4003:S
4004:D 1 import gle_types.
4005:S
4006:D 1 export
4007:S
4008:D 1 var font ['GLE_GLE_STROKE_TABLE'] : array [0..maxint] of gle_shortint;

```

```

4008:S
4010:D 1 procedure gle_soft_char_size ( gcb : graphics_control_block_ptr );
4011:D 1 procedure gle_soft_text_spacing ( gcb : graphics_control_block_ptr );
4012:D 1 procedure gle_soft_text_dir ( gcb : graphics_control_block_ptr );
4013:D 1 procedure gle_soft_text_just ( gcb : graphics_control_block_ptr );
4014:D 1 procedure gle_text_xform ( gcb : graphics_control_block_ptr );
4015:S
4016:D 1 implement
4017:S
4018:D 1 procedure gle_text_xform ( gcb : graphics_control_block_ptr );
4019:S
4020:D 2 const
4021:D 2 fraction_adjust = 32768;
4022:D 2 res_adjust = 8;
4023:D 2 adjusted_cell_width = 7 * fraction_adjust;
4024:D 2 adjusted_cell_height = 9 * fraction_adjust;
4025:S
4026:D -4 2 var i : integer;
4027:D -12 2 dx,dy : integer;
4028:S
4029:C 2 begin
4030:C 2 with gcb^ do
4031:C 3 begin
4032:C 3 $RANGE ON$
4033:C 3 for i := 0 to 15 do
4034:C 4 begin
4035:C 4 if i < 8 then
4036:C 5 begin
4037:C 5 dx := ((i-1) * char_width) div res_adjust;
4038:C 5 cosx_table[i] := (dx * text_cos_dir) div adjusted_cell_width;
4039:C 5 sinx_table[i] := (dx * text_sin_dir) div adjusted_cell_width;
4040:C 5 end;
4041:C 5 dy := ((i-4) * char_height) div res_adjust;
4042:C 5 siny_table[i] := (-dy * text_sin_dir) div adjusted_cell_height;
4043:C 5 cosy_table[i] := (dy * text_cos_dir) div adjusted_cell_height;
4044:C 5 end;
4045:S
4046:C 3 dx := (char_width + char_space);
4047:C 3 text_space_x := (dx * text_cos_dir) div fraction_adjust;
4048:C 3 text_space_y := (dx * text_sin_dir) div fraction_adjust;
4049:C 3
4050:C 3 dy := -(char_height + line_space);
4051:C 3 text_line_x := (-dy * text_sin_dir) div fraction_adjust;
4052:C 3 text_line_y := (dy * text_cos_dir) div fraction_adjust;
4053:C 3 $RANGE OFF$ ( [ added to avoid random vectors )
4054:C 3 calc_text_xform := 0; ( transformation is not needed )
4055:C 3 end;
4056:C 2 end;
4057:S
4058:D 1 procedure gle_soft_char_size ( gcb : graphics_control_block_ptr );
4059:S
4060:C 2 begin
4061:C 2 with gcb^ do
4062:C 3 if (char_width <> info1) or (char_height <> info2) then
4063:C 4 begin
4064:C 4 calc_text_xform := 1; ( calc new transform )
4065:C 4 char_width := info1;
4066:C 4 char_height := info2;
4067:C 4 end;
4068:C 2 end

```



```
4089:S
4070:D      1 procedure gle_soft_text_spacing ( gcb : graphics_control_block_ptr);
4071:S
4072:C      2 begin
4073:C      2   with gcb^ do
4074:C      3     if (char_space <> info1) or (line_space <> info2) then
4075:C      4       begin
4076:C      4         calc_text_xform := 1; { calc new transform }
4077:C      4         char_space := info1;
4078:C      4         line_space := info2;
4079:C      4       end;
4080:C      2 end;
4081:S
4082:D      1 procedure gle_soft_text_dir      ( gcb : graphics_control_block_ptr);
4083:S
4084:C      2 begin
4085:C      2   with gcb^ do
4086:C      3     if (text_cos_dir <> info1) or (text_sin_dir <> info2) then
4087:C      4       begin
4088:C      4         calc_text_xform := 1; { calc new transform }
4089:C      4         text_cos_dir := info1;
4090:C      4         text_sin_dir := info2;
4091:C      4       end;
4092:C      2 end;
4093:S
4094:D      1 procedure gle_soft_text_just      ( gcb : graphics_control_block_ptr);
4095:S
4096:C      2 begin
4097:C      2   with gcb^ do
4098:C      3     if (char_just_x <> info1) or (char_just_y <> info2) then
4099:C      4       begin
4100:C      4         calc_text_xform := 1; { calc new transform }
4101:C      4         char_just_x := info1;
4102:C      4         char_just_y := info2;
4103:C      4       end;
4104:C      2 end;
4105:S
4106:C      1 end. { module gle_stext }
4107:S
4108:S
```

No errors. No warnings.

***** Nonstandard language features enabled *****

GLE_TYPES

Description

GLE_TYPES defines the input and output graphics control blocks (GCBs) used by the GLE routines as well as the machine-type identifier tokens, and some globals for bit-mapped display use.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      0 {
2:D      0 { Graphics Low End
3:D      0 {
4:D      0 {
5:D      0 { Module = GLE_TYPES
6:D      0 { Programmer = BJS
7:D      0 { Date = 10- 5-82
8:D      0 {
9:D      0 { Purpose: To define the graphics control blocks used by GLE.
10:S     0 {
11:D     0 { Rev history
12:D     0 { Created = 10- 5-82 BJS
13:D     0 { Modified = 11-29-82 BJS Clean up and changed booleans to shortints
14:S     0 {
15:S     ( (c) Copyright Hewlett-Packard Company, 1983.
16:S     All rights are reserved. Copying or other
17:S     reproduction of this program except for archival
18:S     purposes is prohibited without the prior
19:S     written consent of Hewlett-Packard Company.
20:S
21:S
22:S     RESTRICTED RIGHTS LEGEND
23:S
24:S     Use, duplication, or disclosure by the Government
25:S     is subject to restrictions as set forth in
26:S     paragraph (b) (3) (B) of the Rights in Technical
27:S     Data and Computer Software clause in
28:S     DAR 7-104.9(a).
29:S
30:S     HEWLETT-PACKARD COMPANY
31:D     0 Fort Collins, Colorado
32:S
33:S     0 $ables on$
34:D     0 $modcal$
35:S     0 $include 'OPTIONS'$ { compiler options }
36:S     { This include file specifies range checking, debug and other compiler
37:D     0 options for the graphics library
38:S
39:D     0 $debug OFF$
40:D     0 $range OFF$
41:D     0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
42:D     0 $FLOAT_HOW TEST$
43:S
44:S
45:S
46:S
47:D     0 $include 'OPTIONS'$ { compiler options }
48:D     0 $linenum 1000$
1000:D
1001:S
1002:D     0 module gle_types;
1003:S
1004:D     1 export
1005:S
1006:D     1 type gle_byte = -128 .. 127;
1007:D     1 gle_shortint = -32768 .. 32767;
1008:S
1009:D     1 anychar = packed array [1..maxint] of char;
1010:D     1 anychar_ptr = ^anychar;
1011:S

```

```

1012:D     1 gle_char6 = packed array [1..6] of char;
1013:S
1014:D     1 graphics_control_block_ptr = ^graphics_control_block;
1015:D     1 graphics_input_control_block_ptr = ^graphics_input_control_block;

```



```

1016:D 1 $page$
1017:D 1 {***** OUTPUT *****}
1018:S
1019:D 1 graphics_control_block =
1020:D 1 packed_record
1021:D 1 info1 : integer; ( holds info passed to and from GLE )
1022:D 1 info2 : integer;
1023:D 1 info3 : integer;
1024:D 1 info4 : integer;
1025:S
1026:D 1 info_ptr1 : anyptr;
1027:D 1 info_ptr2 : anyptr;
1028:S
1029:D 1 await_blanking : procedure ( gcb : graphics_control_block_ptr );
1030:D 1 buffer_mode : procedure ( gcb : graphics_control_block_ptr );
1031:D 1 char_size : procedure ( gcb : graphics_control_block_ptr );
1032:D 1 clear : procedure ( gcb : graphics_control_block_ptr );
1033:D 1 clip_limits : procedure ( gcb : graphics_control_block_ptr );
1034:D 1 cursor : procedure ( gcb : graphics_control_block_ptr );
1035:D 1 define_color_map : procedure ( gcb : graphics_control_block_ptr );
1036:D 1 define_drawing_mode : procedure ( gcb : graphics_control_block_ptr );
1037:D 1 draw : procedure ( gcb : graphics_control_block_ptr );
1038:D 1 fill_index_color : procedure ( gcb : graphics_control_block_ptr );
1039:D 1 flush_buffer : procedure ( gcb : graphics_control_block_ptr );
1040:D 1 get_color_map : procedure ( gcb : graphics_control_block_ptr );
1041:D 1 get_raster : procedure ( gcb : graphics_control_block_ptr );
1042:D 1 get_polygon_info : procedure ( gcb : graphics_control_block_ptr );
1043:D 1 gload : procedure ( gcb : graphics_control_block_ptr );
1044:D 1 graphics_on_off : procedure ( gcb : graphics_control_block_ptr );
1045:D 1 gstore : procedure ( gcb : graphics_control_block_ptr );
1046:D 1 index_color : procedure ( gcb : graphics_control_block_ptr );
1047:D 1 inq_plp2 : procedure ( gcb : graphics_control_block_ptr );
1048:D 1 linewidth : procedure ( gcb : graphics_control_block_ptr );
1049:D 1 linestyle : procedure ( gcb : graphics_control_block_ptr );
1050:D 1 marker : procedure ( gcb : graphics_control_block_ptr );
1051:D 1 marker_size : procedure ( gcb : graphics_control_block_ptr );
1052:D 1 move : procedure ( gcb : graphics_control_block_ptr );
1053:D 1 output_escape1 : procedure ( gcb : graphics_control_block_ptr );
1054:D 1 output_escape0 : procedure ( gcb : graphics_control_block_ptr );
1055:D 1 polygon : procedure ( gcb : graphics_control_block_ptr );
1056:D 1 set_marker : procedure ( gcb : graphics_control_block_ptr );
1057:D 1 text : procedure ( gcb : graphics_control_block_ptr );
1058:D 1 text_dir : procedure ( gcb : graphics_control_block_ptr );
1059:D 1 text_just : procedure ( gcb : graphics_control_block_ptr );
1060:D 1 text_spacing : procedure ( gcb : graphics_control_block_ptr );
1061:S
1062:D 1 dummy_xxx : procedure ( gcb : graphics_control_block_ptr );
1063:S
1064:D 1 io_inq_timeout : procedure ( anyvar iocb_ptr : anyptr; var value : integer );
1065:D 1 io_read : procedure ( anyvar iocb_ptr, dev_buf_ptr : anyptr );
1066:D 1 io_set_timeout : procedure ( anyvar iocb_ptr, anyptr; value : integer );
1067:D 1 io_term : procedure ( anyvar iocb_ptr : anyptr );
1068:D 1 io_write : procedure ( anyvar iocb_ptr, dev_buf_ptr : anyptr );
1069:S
1070:D 1 iocb : anyptr; ( ptr to io system dependent info )
1071:D 1 device_buf : anyptr; ( ptr to device dependent buffer )
1072:D 1 dev_dep_stuff : anyptr; ( ptr to device dependent information )
1073:D 1 device_info : anyptr; ( pointer to device identifier (e.g. 3) )
1074:D 1 device_info_char_count : gle_shortint; ( # char in device_info )
1075:D 1 error_return : gle_shortint; ( used by some procedures )

```

```

1076:D 1 spooling : gle_shortint; ( '1' if output goes to a file )
1077:S
1078:D 1 display_name : gle_char6; ( name of display device )
1079:D 1 display_name_char_count : gle_shortint; ( # chars used in display_name )
1080:D 1 display_handler_name : gle_char6; ( 'HPGL', 'RASTER', 'GPIS' )
1081:D 1 display_handler_char_count : gle_shortint; ( # chars used in display_handler_name )
1082:S
1083:D 1 display_res_x : real; ( points per mm in x direction )
1084:D 1 display_res_y : real; ( points per mm in y direction )
1085:D 1 display_min_x : integer; ( minimum x device coordinate )
1086:D 1 display_min_y : integer; ( minimum y device coordinate )
1087:D 1 display_max_x : integer; ( maximum x device coordinate )
1088:D 1 display_max_y : integer; ( maximum y device coordinate )
1089:S
1090:D 1 { general info stuff }
1091:D 1 background : gle_shortint; ( '1' if drawing in color 0 supported, '0' if not )
1092:D 1 complement_support : gle_shortint; ( '1' if complement drawing supported )
1093:D 1 non_dominant_support : gle_shortint; ( '1' if non-dominant drawing supported )
1094:D 1 erase_support : gle_shortint; ( '1' if erase drawing supported )
1095:D 1 color_map_support : gle_shortint; ( '1' if color map supported )
1096:D 1 polygon_support : gle_shortint; ( '1' if polygons are supported )
1097:D 1 redraw_background : gle_shortint; ( '1' if color 0 can be changed )
1098:D 1 polygon_fill_factor : gle_shortint; ( approx size of a line in device units )
1099:D 1 polygon_solid_fill : gle_shortint; ( fill line spacing which is used for doild fill )
1100:D 1 dither_support : gle_shortint; ( '1' if polygons use dither )
1101:D 1 palette : integer; ( number of distinct colors supported )
1102:D 1 gamut : integer; ( number of colors which can appear at same time )
1103:D 1 cont_linestyles : gle_shortint; ( number of continuous linestyles supported )
1104:D 1 vect_linestyles : gle_shortint; ( number of vector adjusted linestyles supported )
1105:D 1 linewidths : gle_shortint; ( number of linewidths supported )
1106:D 1 char_sizes : gle_shortint; ( number of character sizes supported, -1 for continuously varying sizes )
1107:D 1
1108:S
1109:D 1 { current value information }
1110:D 1 current_pos_x : integer; ( last point used )
1111:D 1 current_pos_y : integer;
1112:D 1 end_x : integer;
1113:D 1 end_y : integer;
1114:S
1115:D 1 marker_type : gle_shortint;
1116:D 1 marker_width : integer;
1117:D 1 marker_height : integer;
1118:S
1119:D 1 kata : gle_shortint; ( '1' if kata char set to be used )
1120:S
1121:D 1 char_width : integer; ( current character info )
1122:D 1 char_height : integer;
1123:D 1 char_space : integer;
1124:D 1 line_space : integer;
1125:D 1 text_dir : integer;
1126:D 1 text_dir_dir : integer;
1127:D 1 char_just_x : integer;
1128:D 1 char_just_y : integer;
1129:S
1130:D 1 clip_limits_xmin : integer;
1131:D 1 clip_limits_xmax : integer;
1132:D 1 clip_limits_ymin : integer;
1133:D 1 clip_limits_ymax : integer;
1134:S
1135:D 1 current_cursor_state : gle_shortint; ( off = 0; on = 1 )

```



```

1136:D 1 current_cursor_x : integer;
1137:D 1 current_cursor_y : integer;
1138:S
1139:D 1 current_buffer_mode : gle_shortint; ( buffered = 1; unbuffered = 0 )
1140:S
1141:D 1 current_linestyle : gle_shortint;
1142:D 1 current_linestyle_pattern : gle_shortint;
1143:D 1 current_pattern_length : gle_shortint;
1144:D 1 current_linestyle_mode : gle_shortint;
1145:S
1146:D 1 current_color_index : gle_shortint;
1147:D 1 current_fill_index : gle_shortint;
1148:D 1 current_drawing_mode : gle_shortint;
1149:D 1 current_linewidth : gle_shortint;
1150:S
1151:D 1 current_polygon_color : gle_shortint; ( index, -1 if rgb )
1152:D 1 current_polygon_red : gle_shortint;
1153:D 1 current_polygon_green : gle_shortint;
1154:D 1 current_polygon_blue : gle_shortint;
1155:S
1156:D 1 { temp storage of some regs }
1157:D 1 old_a5 : integer;
1158:D 1 old_a6 : integer;
1159:S
1160:D 1 { software text stuff }
1161:D 1 text_space_x : integer; ( spacing x&y after characters )
1162:D 1 text_space_y : integer;
1163:S
1164:D 1 text_line_x : integer; ( linefeed spacing )
1165:D 1 text_line_y : integer;
1166:D 1 cosx_table : packed array [0..7] of gle_shortint;
1167:D 1 cosy_table : packed array [0..15] of gle_shortint;
1168:D 1 sinx_table : packed array [0..7] of gle_shortint;
1169:D 1 siny_table : packed array [0..15] of gle_shortint;
1170:S
1171:D 1 calc_text_xform : gle_shortint; ( '1' if text xform needs to be recalculated )
1172:D 1 calc_soft_text_xform : procedure ( gcb : graphics_control_block_ptr );
1173:S
1174:D 1 soft_font_ptr : anyptr; ( points to font table )
1175:S
1176:D 1 soft_text_temp1 : integer;
1177:D 1 soft_text_temp2 : integer;
1178:S
1179:D 1 { software clipping stuff }
1180:D 1 unclipped_move : procedure ( gcb : graphics_control_block_ptr );
1181:D 1 unclipped_draw : procedure ( gcb : graphics_control_block_ptr );
1182:S
1183:D 1 soft_clip_savex0 : integer;
1184:D 1 soft_clip_savex1 : integer;
1185:D 1 soft_clip_savey0 : integer;
1186:D 1 soft_clip_savey1 : integer;
1187:D 1 soft_clip_switch : gle_shortint;
1188:S
1189:D 1 soft_clip_cpx : integer;
1190:D 1 soft_clip_cpy : integer;
1191:D 1 end;

```

```

1192:D 1 $pag:$
1193:D 1 {***** INPUT *****}
1194:S
1195:D 1 graphics_input_control_block =
1196:D 1 packed record
1197:D 1 info1 : integer; ( holds info passed to and from GLE )
1198:D 1 info2 : integer;
1199:D 1 info3 : integer;
1200:D 1 info4 : integer;
1201:S
1202:D 1 info_ptr1 : anyptr;
1203:S
1204:D 1 get_digitize : procedure ( gcb : graphics_input_control_block_ptr );
1205:D 1 inq_pip2 : procedure ( gcb : graphics_input_control_block_ptr );
1206:D 1 input_echo : procedure ( gcb : graphics_input_control_block_ptr );
1207:D 1 input_escape1 : procedure ( gcb : graphics_input_control_block_ptr );
1208:D 1 input_escape0 : procedure ( gcb : graphics_input_control_block_ptr );
1209:D 1 sample : procedure ( gcb : graphics_input_control_block_ptr );
1210:D 1 start_digitize : procedure ( gcb : graphics_input_control_block_ptr );
1211:S
1212:D 1 dummy_xxx : procedure ( gcb : graphics_input_control_block_ptr );
1213:S
1214:D 1 io_inq_timeout : procedure ( anyvar iocb_ptr : anyptr; var value : integer );
1215:D 1 io_read : procedure ( anyvar iocb_ptr, dev_buf_ptr : anyptr );
1216:D 1 io_set_timeout : procedure ( anyvar iocb_ptr : anyptr; value : integer );
1217:D 1 io_term : procedure ( anyvar iocb_ptr : anyptr );
1218:D 1 io_write : procedure ( anyvar iocb_ptr, dev_buf_ptr : anyptr );
1219:S
1220:D 1 iocb : anyptr; ( ptr to io system dependent info )
1221:D 1 device_buf : anyptr; ( ptr to device dependent buffer )
1222:D 1 dev_dep_stuff : anyptr; ( ptr to device dependent information )
1223:D 1 device_info : anyptr; ( pointer to device identifier (e.g. 701) )
1224:D 1 device_info_char_count : gle_shortint; ( # char in device_info )
1225:D 1 error_return : gle_shortint; ( used by some procedures )
1226:S
1227:D 1 input_name : gle_char6; ( name of display device )
1228:D 1 input_name_char_count : gle_shortint; ( # chars used in display_name )
1229:D 1 input_handler_name : gle_char6; ( 'HPGL', 'KN08', 'GPIS' )
1230:D 1 input_handler_char_count : gle_shortint; ( # chars used in display_handler_name )
1231:S
1232:D 1 input_res_x : real; ( points per mm in x direction )
1233:D 1 input_res_y : real; ( points per mm in y direction )
1234:D 1 input_min_x : integer; ( input, min and max device coordinates )
1235:D 1 input_max_x : integer;
1236:D 1 input_min_y : integer;
1237:D 1 input_max_y : integer;
1238:S
1239:D 1 input_cpx : integer; ( Where device dependent echoes are started )
1240:D 1 input_cpy : integer;
1241:D 1 end;
1242:D 1
1243:D 1 status_def =
1244:D 1 packed record
1245:D 1 bit15, bit14, bit13, bit12, bit11, bit10, bit9, bit8,
1246:D 1 notbusy, bit6, vblank, bit4, hz50, monochrome, bit1, bit0 : boolean;
1247:D 1 end;
1248:S
1249:D 1 w_array = packed array [1..maxint] of gle_shortint;
1250:S
1251:S

```



```

1252:D      1 const
1253:D      1 munknown = 0;
1254:D      1 m9818a = 1;
1255:D      1 m9826a = 2;
1256:D      1 m9836a = 3;
1257:D      1 m9836c = 4;
1258:D      1 m98627a = 5;
1259:D      1 m9837a = 6;
1260:D      1 mgator_c = 7;
1261:S
1262:S
1263:D      1 no = 0;
1264:D      1 yes = 1;
1265:S
1266:D      1 var
1267:S
1268:D      -4 1 rule_reg : ^gle_shortint; { [HEX('564008')] : gle_shortint;}
1269:D      -8 1 width_reg : ^gle_shortint; { [HEX('56400C')] : gle_shortint;}
1270:S      1 inq_buffer : gle_shortint; { [HEX('564100')] : packed array [0..maxint] of
1271:D      -12 1 { gle_shortint;}
1272:D      -16 1 crt : ^w_array; { [HEX('566000')] : word_array;}
1273:D      -20 1 status : ^status_def; { [hex('564000')] : status_def;}
1274:D      -24 1 ww_reg : ^gle_shortint; { [hex('564108')] : gle_shortint;}
1275:D      -24 1 blink1_reg [hex('564100')] : gle_shortint;
1276:D      -24 1 blink2_reg [hex('564104')] : gle_shortint;
1277:D      -25 1 write_to_status : boolean;
1278:D      -30 1 control_space : integer;
1279:S
1280:D      -30 1 implement
1281:S

```

```

1282:D      -30 1 $page$

```

Dump of GLE_TYPES

Imported:

Exported:

```

ANYCHAR      type
  array elbitsize=8 unpacksize=2147483647 align=2
ANYCHAR_PTR  type
  pointer unpacksize=4 align=2
BLINK1_REG   var lev=1 addr=131072 long
BLINK2_REG   var lev=1 addr= 65578 long
CONTROL_SPACE var lev=1 addr=-30 globalbase = GLE_TYPES
CRT          var lev=1 addr= -16 globalbase = GLE_TYPES
GLE_BYTE     type
  subrange min=-128 max=127 unpacksize=2 align=2 bitsize=8 signed
GLE_CHAR6    type
  array elbitsize=8 unpacksize=6 align=2
GLE_SHORTINT type
  subrange min=-32768 max=32767 unpacksize=2 align=2 bitsize=16 signed
GLE_TYPES    module lev=1
GRAPHICS_CONTROL_BLOCK type
  record unpacksize=730 align=2
    WAIT_BLANKING field offset=24
    BACKGROUND    field offset=398
    BUFFER_MODE    field offset=32
    CALC_SOFT_TEXT_XFORM field offset=668
    CALC_TEXT_XFORM field offset=666
    CHAR_HEIGHT    field offset=466
    CHAR_JUST_X     field offset=486
    CHAR_JUST_Y     field offset=490
    CHAR_SIZE       field offset=40
    CHAR_SIZES      field offset=432
    CHAR_SPACE      field offset=470
    CHAR_WIDTH      field offset=462
    CLEAR           field offset=48
    CLIP_LIMITS     field offset=56
    CLIP_LIMITS_XMAX field offset=498
    CLIP_LIMITS_XMIN field offset=494
    CLIP_LIMITS_YMAX field offset=506
    CLIP_LIMITS_YMIN field offset=502
    COLOR_MAP_SUPPORT field offset=406
    COMPLEMENT_SUPPORT field offset=400
    CONT_LINESTYLES field offset=426
    COSX_TABLE      field offset=570
    COSY_TABLE      field offset=586
    CURRENT_BUFFER_MODE field offset=520
    CURRENT_COLOR_INDEX field offset=530
    CURRENT_CURSOR_STATE field offset=510
    CURRENT_CURSOR_X field offset=512
    CURRENT_CURSOR_Y field offset=516
    CURRENT_DRAWING_MODE field offset=534
    CURRENT_FILL_INDEX field offset=532
    CURRENT_LINestyle field offset=522
    CURRENT_LINestyle_MODE field offset=528
    CURRENT_LINestyle_PATTERN field offset=524
    CURRENT_LINewidth field offset=536
    CURRENT_PATTERN_LENGTH field offset=526
    CURRENT_POLYGON_BLUE field offset=544
    CURRENT_POLYGON_COLOR field offset=538

```



```

CURRENT_POLYGON_GREEN field offset=542
CURRENT_POLYGON_RED field offset=540
CURRENT_POS_X field offset=434
CURRENT_POS_Y field offset=438
CURSOR field offset=64
DEFINE_COLOR_MAP field offset=72
DEFINE_DRAWING_MODE field offset=80
DEVICE_BUF field offset=332
DEVICE_INFO field offset=340
DEVICE_INFO_CHAR_COUNT field offset=344
DEV_DEP_STUFF field offset=336
DISPLAY_HANDLER_CHAR_COUNT field offset=364
DISPLAY_HANDLER_NAME field offset=358
DISPLAY_MAX_X field offset=390
DISPLAY_MAX_Y field offset=394
DISPLAY_MIN_X field offset=382
DISPLAY_MIN_Y field offset=386
DISPLAY_NAME field offset=350
DISPLAY_NAME_CHAR_COUNT field offset=356
DISPLAY_RES_X field offset=366
DISPLAY_RES_Y field offset=374
DITHER_SUPPORT field offset=416
DRAW field offset=386
DUMMY_XXX field offset=280
END_X field offset=442
END_Y field offset=446
ERASE_SUPPORT field offset=404
ERROR_RETURN field offset=346
FILL_INDEX_COLOR field offset=96
FLUSH_BUFFER field offset=104
GAMUT field offset=422
GET_COLOR_MAP field offset=112
GET_POLYGON_INFO field offset=128
GET_PASTER field offset=120
GLOAD field offset=136
GRAPHICS_ON_OFF field offset=144
GSTOPE field offset=152
INDEX_COLOR field offset=160
INFO0 field offset=0
INFO1 field offset=4
INFO2 field offset=8
INFO3 field offset=12
INFO4 field offset=16
INFO_PTR1 field offset=20
INFO_PTR2 field offset=168
INQ_PIP2 field offset=328
IOCB field offset=288
IO_INQ_TIMEOUT field offset=296
IO_READ field offset=304
IO_SET_TIMEOUT field offset=312
IO_TERM field offset=320
IO_WRITE field offset=460
KATA field offset=184
LINESTYLE field offset=176
LINEWIDTH field offset=430
LINEWIDTHS field offset=474
LINE_SPACE field offset=192
MARKER field offset=456
MARKER_HEIGHT field offset=200
MARKER_SIZE field offset=450
MARKER_TYPE field offset=450

```

```

MARKER_WIDTH field offset=452
MOVE field offset=208
NON_DOMINANT_SUPPORT field offset=402
OLD_A5 field offset=546
OLD_A6 field offset=550
OUTPUT_ESCAPE1 field offset=216
OUTPUT_ESCAPE0 field offset=224
PALLETTE field offset=418
POLYGON field offset=232
POLYGON_FILL_FACTOR field offset=412
POLYGON_SOLID_FILL field offset=414
POLYGON_SUPPORT field offset=408
REDEF_BACKGROUND field offset=410
SFT_MARKER field offset=240
SINK_TABLE field offset=618
SINY_TABLE field offset=634
SOFT_CLIP_CPX field offset=722
SOFT_CLIP_CPY field offset=726
SOFT_CLIP_SAVEX0 field offset=704
SOFT_CLIP_SAVEX1 field offset=708
SOFT_CLIP_SAVEY0 field offset=712
SOFT_CLIP_SAVEY1 field offset=716
SOFT_CLIP_SWITCH field offset=720
SOFT_FONT_PTR field offset=676
SOFT_TEXT_TEMP1 field offset=680
SOFT_TEXT_TEMP2 field offset=684
SPOOLING field offset=348
TEXT field offset=248
TEXT_COS_DIR field offset=482
TEXT_DIR field offset=256
TEXT_JUST field offset=264
TEXT_LINE_X field offset=562
TEXT_LINE_Y field offset=566
TEXT_SIN_DIR field offset=478
TEXT_SPACE_X field offset=554
TEXT_SPACE_Y field offset=558
TEXT_SPACING field offset=272
UNCLIPPED_DRAW field offset=696
UNCLIPPED_MOVE field offset=688
VECT_LINESTYLES field offset=428
GRAPHICS_CONTROL_BLOCK_PTR type
  pointer unpacksize=4 align=2
GRAPHICS_INPUT_CONTROL_BLOCK type
  record unpacksize=200 align=2
    DEVICE_BUF field offset=128
    DEVICE_INFO field offset=136
    DEVICE_INFO_CHAR_COUNT field offset=140
    DEV_DEP_STUFF field offset=132
    DUMMY_XXX field offset=76
    ERROR_RETURN field offset=142
    GET_DIGITIZE field offset=20
    INFO1 field offset=0
    INFO2 field offset=4
    INFO3 field offset=8
    INFO4 field offset=12
    INFO_PTR1 field offset=16
    INPUT_CPX field offset=192
    INPUT_CPY field offset=196
    INPUT_ECHO field offset=36
    INPUT_ESCAPE1 field offset=44

```



```

INPUT_ESCAPED      field offset=52
INPUT_HANDLER_CHAR_COUNT field offset=158
INPUT_HANDLER_NAME field offset=152
INPUT_MAX_X        field offset=180
INPUT_MAX_Y        field offset=188
INPUT_MIN_X        field offset=176
INPUT_MIN_Y        field offset=184
INPUT_NAME         field offset=144
INPUT_NAME_CHAR_COUNT field offset=150
INPUT_RES_X        field offset=160
INPUT_RES_Y        field offset=168
INQ_PIP2           field offset=28
IOCB               field offset=124
IO_INQ_TIMEOUT     field offset=84
IO_READ            field offset=92
IO_SET_TIMEOUT     field offset=100
IO_TERM            field offset=108
IO_WRITE           field offset=116
SAMPLE             field offset=60
START_DIGITIZE     field offset=68
GRAPHICS_INPUT_CONTROL_BLOCK_PTR type
  pointer unpacksize=4 align=2
INQ_BUFFER          var lev= 1 addr= -12 globalbase = GLE_TYPES
M9816A              konst 1
M9828A              konst 2
M9836A              konst 3
M9836C              konst 4
M9837A              konst 6
M98627A             konst 5
MGATOR_C            konst 7
MUNKNOQN           konst 0
NO                  konst 0
RULE_REG            var lev= 1 addr= -4 globalbase = GLE_TYPES
STATUS              var lev= 1 addr= -20 globalbase = GLE_TYPES
STATUS_DEF          type
  record unpacksize=2 align=2
  BIT0              field offset=0 bitoffset=15
  BIT1              field offset=0 bitoffset=14
  BIT10             field offset=0 bitoffset=5
  BIT11             field offset=0 bitoffset=4
  BIT12             field offset=0 bitoffset=3
  BIT13             field offset=0 bitoffset=2
  BIT14             field offset=0 bitoffset=1
  BIT15             field offset=0 bitoffset=0
  BIT4              field offset=0 bitoffset=11
  BIT6              field offset=0 bitoffset=9
  BIT8              field offset=0 bitoffset=7
  BIT9              field offset=0 bitoffset=6
  H250              field offset=0 bitoffset=12
  MONOCHROME        field offset=0 bitoffset=13
  NOTBUSY            field offset=0 bitoffset=8
  VBLANK            field offset=0 bitoffset=10
WIDTH_REG           var lev= 1 addr= -8 globalbase = GLE_TYPES
WRITE_TO_STATUS     var lev= 1 addr= -25 globalbase = GLE_TYPES
WJ_REG             var lev= 1 addr= -24 globalbase = GLE_TYPES
W_ARRAY            type
  array elbitsize=16 unpacksize=4 (OFLO) align=2
YES                 konst 1

```

GLE_TYPES dump complete

```

1283:C      1 end. ( GLE_TYPES )
1284:D      1 $LIST ON$
1285:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

GLE_UTLS

Description

GLE_UTLS provides general tools for GLE routines.

Requirements

GLE_TYPES.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:D      0 { Graphics Low End }
2:D      0 {
3:D      0 {
4:D      0 { Module = GLE_UTILS
5:D      0 { Programmer = BJS
6:D      0 { Date = 11-15-82
7:D      0 {
8:D      0 { Purpose: To provide general GLE tools
9:S
10:D     0 { Rev history
11:D     0 { Created - 11-15-82
12:D     0 { Modified - XX-XX-XX
13:S
14:S      (c) Copyright Hewlett-Packard Company, 1983.
15:S      All rights are reserved. Copying or other
16:S      reproduction of this program except for archival
17:S      purposes is prohibited without the prior
18:S      written consent of Hewlett-Packard Company.
19:S
20:S
21:S      RESTRICTED RIGHTS LEGEND
22:S
23:S      Use, duplication, or disclosure by the Government
24:S      is subject to restrictions as set forth in
25:S      paragraph (b) (3) (8) of the Rights in Technical
26:S      Data and Computer Software clause in
27:S      DAR 7-104.9(a).
28:S
29:S      HEWLETT-PACKARD COMPANY
30:D     0      Fort Collins, Colorado
31:S
32:D     0 $search 'GLE_TYPES'$
33:D     0 $sysprogs
34:D     0 $include 'OPTIONS'$ ( ***** COMPILER OPTIONS ***** )
35:S      ( This include file specifies range checking, debug and other compiler
36:D     0      options for the graphics library
37:S
38:D     0 $debug OFF$
39:D     0 $range OFF$
40:D     0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
41:D     0 $FLOAT_HDW TEST$
42:S
43:S
44:S
45:S
46:D     0 $include 'OPTIONS'$ ( ***** COMPILER OPTIONS ***** )
47:D     0 $linenum 21000$
21000:D
21001:S
21002:D     0 module gle_utils;
21003:S
21004:D     1 import gle_types;
21005:S
21006:D     1 export
21007:S
21008:D     1 function gle_read_integer ( size : gle_shortint; s : anychar_ptr; var count : gle_shortint ) : gle_
shortint;
21009:D     1 procedure gle_write_integer ( value : gle_shortint; var size : gle_shortint; s : anychar_ptr );
21010:D     1 procedure gle_copy_to_string ( s : anychar_ptr; sl : gle_shortint; var d : string );
21011:D     1 function gle_match ( size1 : gle_shortint; sl : anychar_ptr;

```

```

21012:D     2      size2 : gle_shortint; s2 : anychar_ptr ) : boolean;
21013:D     1 function gle_shortint_min ( p1,p2 : gle_shortint ) : gle_shortint;
21014:D     1 function gle_shortint_max ( p1,p2 : gle_shortint ) : gle_shortint;
21015:S

```



```

21016:D      1 $PAGES
21017:D      1 implement
21018:S
21019:D      1 function gle_shortint_min ( p1,p2 : gle_shortint ) : gle_shortint;
21020:S
21021:D      2 { Purpose : To return the minimum 16 bit integer }
21022:S
21023:D      2 begin
21024:C      2   if p1 < p2 then gle_shortint_min := p1
21025:C      2   else gle_shortint_min := p2;
21026:C      2 end; { gle_shortint_min }
21027:D
21028:D      1 function gle_shortint_max ( p1,p2 : gle_shortint ) : gle_shortint;
21029:S
21030:D      2 { Purpose : To return the maximum 16 bit integer }
21031:S
21032:D      2 begin
21033:C      2   if p1 > p2 then gle_shortint_max := p1
21034:C      2   else gle_shortint_max := p2;
21035:C      2 end; { gle_shortint_max }
21036:D
21037:D      1 procedure gle_write_integer ( value : gle_shortint;
21038:D      2                               var size : gle_shortint;
21039:D      2                               s : anychar_ptr );
21040:S
21041:D      2 { Purpose : To convert a 16 bit signed integer to ASCII }
21042:S
21043:D      2 var
21044:D      2   t : packed array [1..20] of char;
21045:D      2   i : gle_shortint;
21046:D      2   temp_value : gle_shortint;
21047:D      2   digit : gle_shortint;
21048:S
21049:D      2 begin
21050:D      2   temp_value := abs(value);
21051:D      2   i := 0;
21052:D      2   repeat
21053:D      2     i := i + 1;
21054:D      2     digit := temp_value mod 10;
21055:D      2     t[i] := chr((digit) + 48);
21056:D      2     temp_value := temp_value div 10;
21057:D      2   until temp_value = 0;
21058:D      2   if value < 0 then
21059:D      2     begin
21060:D      2       i := i + 1;
21061:D      2       t[i] := '-';
21062:D      2     end;
21063:D      2   size := i;
21064:D      2   for i := 1 downto 1 do
21065:D      2     s[size-i+1] := t[i];
21066:D      2   end; { write_integer }
21067:D
21068:D      1 function gle_read_integer( size : gle_shortint; s : anychar_ptr; var count : gle_shortint ) : gle_sh
21069:D      2   ortint;
21070:S
21071:D      2 { Purpose : To convert from ASCII to a 16 bit signed integer }
21072:S
21073:D      2 var
21074:D      2   value : gle_shortint;

```

```

21075:D      -3 2   neg : boolean;
21076:D      -6 2   digit : gle_shortint;
21077:D      -8 2   i : gle_shortint;
21078:D      -10 2  start : gle_shortint;
21079:S
21080:D      2 begin
21081:D      2   i := 1;
21082:D      2   while (s[i] = ' ') and ( i <= size ) do i := i + 1;
21083:D      2   if i > size then escape(-8);
21084:S
21085:D      2   if s[i] = '-' then
21086:D      2     begin
21087:D      2       i := i + 1;
21088:D      2       neg := true;
21089:D      2     end
21090:D      2   else
21091:D      2     neg := false;
21092:D      2     value := 0;
21093:D      2     start := i;
21094:D      2     while (s[i] >= '0') and (s[i] <= '9') and (i <= size) do
21095:D      2       begin
21096:D      2         digit := ord(s[i]) - 48;
21097:D      2         value := value * 10 + digit;
21098:D      2         i := i + 1;
21099:D      2       end;
21100:D      2   if (i > size+1) or (i = start) then escape(-8);
21101:D      2   count := i; { next free byte }
21102:D      2   if neg then value := -value;
21103:D      2   gle_read_integer := value;
21104:D      2 end; { read_integer }
21105:S
21106:D      1 procedure gle_copy_to_string ( s : anychar_ptr; s1 : gle_shortint; var d : string );
21107:S
21108:D      2 { Purpose : To convert from a packed array of char to string format }
21109:S
21110:D      -2 2 var i : gle_shortint;
21111:S
21112:D      2 begin
21113:D      2   setstrlen(d,s1);
21114:D      2   for i := 1 to s1 do
21115:D      2     d[i] := s[i];
21116:D      2 end; { copy_to_string }
21117:S
21118:D      1 function gle_match ( size1 : gle_shortint; s1 : anychar_ptr;
21119:D      2                       size2 : gle_shortint; s2 : anychar_ptr ) : boolean;
21120:S
21121:D      2 { Purpose : To return true if the two packed array of char match }
21122:S
21123:D      -2 2 var i : gle_shortint;
21124:S
21125:D      2 begin
21126:D      2   if size1 = size2 then
21127:D      2     begin
21128:D      2       i := 1;
21129:D      2       $partial_eval on $
21130:D      2       while (i <= size1) and (s1[i] = s2[i]) do i := i + 1;
21131:D      2       gle_match := i > size1;
21132:D      2     end
21133:D      2   else gle_match := false;
21134:D      2 end; { match }

```


21135:S
21136:C 1 end.
21137:S

No errors. No warnings.
***** Nonstandard language features enabled *****

H_DRV

Description

H_DRV provides low-level driver support.

Usage

H_DRV is used by the 98624 and built-in HP-IB interfaces.

Requirements

SYSGLOBALS and IODECLARATIONS.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*)
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1983.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S     RESTRICTED RIGHTS LEGEND
11:S
12:S     Use, duplication, or disclosure by the Government
13:S     is subject to restrictions as set forth in
14:S     paragraph (b) (3) (8) of the Rights in Technical
15:S     Data and Computer Software clause in
16:S     DAR 7-104.9(a).
17:S
18:S     HEWLETT-PACKARD COMPANY
19:S     0 Fort Collins, Colorado      *)
20:S
21:S
22:D     0 $MODCAL ON$
23:D     0 $PARTIAL EVAL ON$
24:D     0 $STACKCHECK ON$
25:D     0 $RANGE OFF$
26:D     0 $DEBUG OFF$
27:D     0 $OVFLCHECK OFF$

```

```

28:D     0 $PAGES$
29:S
30:D     0 {*****}
31:D     0 {*}
32:D     0 {*      not RELEASED      VERSION      3.0}
33:D     0 {*}
34:D     0 {*****}
35:D     0 {*}
36:D     0 {*}
37:D     0 {*      IOLIB      HP1B_DRIVERS}
38:D     0 {*}
39:D     0 {*}
40:D     0 {*****}
41:D     0 {*}
42:D     0 {*}
43:D     0 {*      library      - IOLIB}
44:D     0 {*      name      - HP1B_DRIVERS}
45:D     0 {*      module(s) - h_drv}
46:D     0 {*      - hp1b}
47:D     0 {*}
48:D     0 {*      author      -}
49:D     0 {*      phone      -}
50:D     0 {*}
51:D     0 {*      date      - June 1 , 1981}
52:D     0 {*      update     - Aug 1 , 1983}
53:D     0 {*      release    - ?????????}
54:D     0 {*}
55:D     0 {*      source      - IOLIB:H_DRV.TEXT}
56:D     0 {*      object      - IOLIB:H_DRV.CODE}
57:D     0 {*}
58:D     0 {*****}

```



```
59:D 0 $PAGE$
60:D 0 (*****
61:D 0 (*
62:D 0 (*
63:D 0 (*      BUG FIX HISTORY      - after release
64:D 0 (*
65:D 0 (*
66:D 0 (*      BUG #      8Y / ON      LOC      DESCRIPTION
67:D 0 (*      -----
68:D 0 (*
69:D 0 (*      367      -----      hpib_initialize Allow eXecute of driver
70:D 0 (*      09/22/82      and have it install
71:D 0 (*      itself in the system.
72:D 0 (*
73:D 0 (*****
74:S
```

```
75:D 0 $PAGE$
76:D 0 (*****
77:D 0 (*
78:D 0 (*
79:D 0 (*      This is the source code for an external procedures library
80:D 0 (*      to be used for general purpose interfacing on the HP 9826.
81:D 0 (*
82:D 0 (*      The library consists of 3 primary sets of modules -
83:D 0 (*
84:D 0 (*          1.      KERNEL modules
85:D 0 (*          2.      driver modules
86:D 0 (*          3.      IOLIB modules
87:D 0 (*
88:D 0 (*      The KERNEL modules consist of the following modules -
89:D 0 (*
90:D 0 (*          1.      iodeclarations ( contains static r/w space )
91:D 0 (*          2.      iocomasm
92:D 0 (*          3.      general_0      ( initialization & low level
93:D 0 (*          routines like ioread/iowrite)
94:D 0 (*
95:D 0 (*      The KERNEL modules also have an executable program segment
96:D 0 (*      that gets executed at the time it is loaded. This program
97:D 0 (*      initializes the static read/write memory. This program also
98:D 0 (*      allocates the temporary storage for any card that exists -
99:D 0 (*      independent of whether there is or is not a driver for it.
100:D 0 (*
101:D 0 (*      The driver modules consist of the actual assembly or PASCAL
102:D 0 (*      routines that deal with a specific interface card. There is
103:D 0 (*      also an executable program segment for each driver module.
104:D 0 (*      This program searches the select code table in the static r/w
105:D 0 (*      initialized by the KERNEL general_0 module for all select codes
106:D 0 (*      that have the right interface card ( HPiB drivers will search
107:D 0 (*      for the 98624 interface ). This program will then set up the
108:D 0 (*      driver tables to point to the correct drivers.
109:D 0 (*
110:D 0 (*      The rest of the IDLIB modules are high-level modules that are
111:D 0 (*      used by an end user in his/her application program.
112:D 0 (*
113:D 0 (*      The KERNEL and some set of driver modules will exist in the
114:D 0 (*      SYSTEM.INITLIB file as object code (not EXPORT text ). The
115:D 0 (*      export text will reside on the SYSTEM.LIBRARY file. The rest
116:D 0 (*      of the library will reside on the SYSTEM.LIBRARY.
117:D 0 (*****
```



```
118:D 0 $PAGE$
119:D 0 {*****}
120:D 0 {*}
121:D 0 {*}
122:D 0 {* REFERENCE$ :}
123:D 0 {*}
124:D 0 {*}
125:D 0 {* 1. 9826 I/O Designers Guide ( ----- )}
126:D 0 {*}
127:D 0 {* 2. 68000 Manual ( Motorola )}
128:D 0 {*}
129:D 0 {* 3. Pascal alpha site ERS ( ----- )}
130:D 0 {*}
131:D 0 {* 4. Pascal I/O Library ERS ( ----- )}
132:D 0 {*}
133:D 0 {* 5. 9826 HPL EIO & IOD listings ( ----- )}
134:D 0 {*}
135:D 0 {* 6. 9826 HPL Misc. I/O Doc. ( ----- )}
136:D 0 {*}
137:D 0 {* 7. 9826 card documentation ( Mfg. Specs. )}
138:D 0 {*}
139:D 0 {* 8. Pascal I/O Library IRS ( ----- )}
140:D 0 {*}
141:D 0 {*****}
142:D 0 {*****}
```

```
143:D 0 $PAGE$
144:D 0 PROGRAM hpib_initialize ( INPUT , OUTPUT );
145:D 0
146:D 0 ( This module has a program segment so that there is
147:D 0 an executable entry point into the module.
148:D 0 At INITLIB time this program is executed. )
149:D 1
```



```
150:D 1 $PAGES
151:D 1 EXTERNAL MODULE exh;
152:S {
153:S   date 08/25/81
154:S   update 10/02/81
155:S
156:S   purpose This module is a declaration of the importation text for
157:S           the external drivers.
158:S
159:S   note The assembly language code that is imported needs to be
160:S        called 'exh'. The routines need to be called
161:S        'exh_@00000' - eh_init referenced below would be
162:S        exh_eh_init. 'eh' refers to external HP-IB.
163:D 1
164:S
165:D 1 IMPORT $SEARCH 'IOLIB:KERNEL.CODE'$
166:D 1 sysglobals, iodeclarations;
167:S
168:D 1 EXPORT
169:S
170:D 1 PROCEDURE eh_init ( temp : ANYPTR );
171:D 1 PROCEDURE eh_isr ( temp : PISRIB );
172:D 1 PROCEDURE eh_rdb ( temp : ANYPTR; VAR x : CHAR );
173:D 1 PROCEDURE eh_wtb ( temp : ANYPTR; val : CHAR );
174:D 1 PROCEDURE eh_rdw ( temp : ANYPTR; VAR x : io_word );
175:D 1 PROCEDURE eh_wtw ( temp : ANYPTR; val : io_word );
176:D 1 PROCEDURE eh_rds ( temp : ANYPTR; VAR x : io_word );
177:D 2
178:D 1 PROCEDURE eh_wtc ( temp : ANYPTR; reg : io_word;
179:D 2 val : io_word );
180:D 1 PROCEDURE eh_tfr ( temp : ANYPTR; bcb : ANYPTR );
181:D 1 PROCEDURE eh_send ( temp : ANYPTR; val : CHAR );
182:D 1 PROCEDURE eh_ppoll ( temp : ANYPTR; VAR x : CHAR );
183:D 1 PROCEDURE eh_clr ( temp : ANYPTR; line : io_bit );
184:D 1 PROCEDURE eh_set ( temp : ANYPTR; line : io_bit );
185:D 1 PROCEDURE eh_test ( temp : ANYPTR; line : io_bit );
186:D 2
187:D 1 PROCEDURE eh_end ( temp : ANYPTR; VAR x : BOOLEAN );
188:S
189:D 1 END; { of exh }
```

```
190:D 1 $PAGES
191:S
192:S
193:D 1 MODULE init_hpib;
194:S {
195:S   date 08/25/81
196:S   update 12/16/82
197:S   8/01/83 JS change rev number
198:S
199:S   purpose This module initializes the HP-IB drivers.
200:S
201:D 1
202:S
203:D 1 IMPORT iodeclarations;
204:S
205:S
206:D 1 EXPORT
207:D 1 VAR
208:D -120 1 hpib_drivers : drv_table_type;
209:S
210:S
211:D -120 1 PROCEDURE io_init_hpib;
212:S
213:S
214:D -120 1 IMPLEMENT
215:D -120 1
216:D -120 1 IMPORT sysglobals,
217:D -120 1 isr,
218:D -120 1 genefal_0,
219:D -120 1 exh;
220:S
221:S
222:D 1 PROCEDURE io_init_hpib;
223:D 2
224:D -2 2 VAR
225:D -6 2 io_isc : type_isc;
226:D -8 2 dummy : INTEGER;
227:D 2 io_lvl : io_byte;
228:S
229:D 2 BEGIN
230:D 2
231:D 2 io_revid := io_revid + ' H3.0'; ( HP-IB revision added 2/5/82 TM )
232:D 2
233:D 2 { set up the driver tables }
234:D 2
235:D 2 WITH hpib_drivers DO BEGIN
236:D 3 hpib_drivers := dummy_drivers;
237:D 3
238:D 3 ioc_init := eh_init;
239:D 3 ioc_isr := eh_isr;
240:D 3 ioc_rdb := eh_rdb;
241:D 3 ioc_wtb := eh_wtb;
242:D 3 ioc_rdw := eh_rdw;
243:D 3 ioc_wtw := eh_wtw;
244:D 3 ioc_rds := eh_rds;
245:D 3 ioc_wtc := eh_wtc;
246:D 3 ioc_end := eh_end;
247:D 3 ioc_tfr := eh_tfr;
248:D 3 ioc_send := eh_send;
249:D 3 ioc_ppoll := eh_ppoll;
250:D 3 ioc_set := eh_set;
251:D 3 ioc_clr := eh_clr;
252:D 3 ioc_test := eh_test;
```



```

250:C      3      END; { of WITH }
251:S
252:S
253:C      2      { set up drivers for the interfaces }
254:S
255:C      2      FOR io_isc:=iominisc TO iomaxisc DO
256:C      3      WITH isc_table[io_isc] DO BEGIN
257:S
258:C      4      IF ( card_id = hp98624 ) OR
259:C      5      { card_id = internal_hpib }
260:C      5      THEN BEGIN
261:S
262:C      5      io_drv_ptr:=ADDR(hpib_drivers);
263:S
264:C      5      WITH io_drv_ptr^, io_tmp_ptr^ DO BEGIN
265:C      6      IF io_isc=7
266:C      7      THEN BEGIN
267:C      7      io_lvl := 3;
268:C      7      END
269:C      7      ELSE BEGIN
270:C      7      io_lvl:=(ioread_byte(io_isc,3) DIV 16) MOD 4)+3;
271:C      7      END; { of IF }
272:S
273:C      6      { if the card exists then link in an ISR for it }
274:C      6      { ??? - what happens if an ISR fires during init }
275:S
276:C      6      IF myisrib.INTREGADDR <> NIL
277:C      7      THEN BEGIN
278:C      7      { remove any existant isr }
279:C      7      ISRUNLINK(io_lvl, { interrupt level }
280:C      7      ADDR(myisrib)); { ptr to the isrib }
281:C      7      END; { of IF }
282:S
283:C      6      PERMISRLINK(iod_isr, { isr
284:C      6      ANYPTR(INTEGER(card_ptr)+3), { int reg addr }
285:C      6      192, { int reg mask }
286:C      6      192, { int reg value }
287:C      6      io_lvl, { int level }
288:C      6      ADDR(myisrib)); { isrib info }
289:C      6      END; { of WITH // DO BEGIN }
290:S
291:C      5      END; { of IF card_id }
292:S
293:C      4      END; { of FOR io_isc WITH isc_table[io_isc] BEGIN }
294:S
295:S
296:C      2      { call the actual driver initialization }
297:S
298:S      { this is sepearte from the set up stuff in case
299:S      there are 2 or more cards connected - and generate
300:C      2      an isr }
301:S
302:C      2      FOR io_isc:=iominisc TO iomaxisc DO
303:C      3      WITH isc_table[io_isc] DO
304:C      4      IF ( card_id = hp98624 ) OR ( card_id = internal_hpib )
305:C      5      THEN BEGIN
306:C      5      CALL(io_drv_ptr^.iod_init , io_tmp_ptr);
307:C      5      END; { of WITH IF }
308:S
309:S

```

```

310:C      2      END; { of io_init_hpib }
311:S
312:C      1      END; { of MODULE init_hpib }

```



```
313:D      1 $PAGE$
314:S
315:D      1 IMPORT      init_hpib ,
316:D      1              LOADER ;
317:S
318:S
319:C      1 BEGIN
320:C      1   io_init_hpib;
321:C      1   harkuser;
322:C      1 END. ( of hpib_initialize )
( 367 TM 9/22/82 )
( 367 TM 9/22/82 )
```

No errors. No warnings.

*****Nonstandard language features enabled*****

HEAPT

Description

HEAPT supports the Pascal procedures: NEW, DISPOSE, MARK and RELEASE.

Usage

The Compiler emits calls to the routines when the \$HEAP_DISPOSE\$ directive has been turned on.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1983.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S     RESTRICTED RIGHTS LEGEND
11:S
12:S     Use, duplication, or disclosure by the Government
13:S     is subject to restrictions as set forth in
14:S     paragraph (b) (3) (B) of the Rights in Technical
15:S     Data and Computer Software clause in
16:S     DAR 7-104.9(a).
17:S
18:S     HEWLETT-PACKARD COMPANY
19:D     0 Fort Collins, Colorado      *)
20:S
21:S
22:D     0 (* 3/2/83 - 3:32 pm *)
23:D     0 $modcal,debug off,range off$
24:D     0 $stackcheck off$      (* 2-Mar-83 -- 3:32 pm *)
25:S
26:D     0 external module sysglobals;
27:D     1 export
28:D     1   var
29:D     -2 1   escapeco: -32768..32767;
30:D     -6 1   FIBptr: integer;
31:D     -10 1  tryrec: integer;
32:D     -14 1  heapmax: integer;
33:D     -18 1  heapbase: integer;
34:D     -22 1  iorecresult: integer;
35:D     1 end;
36:S
37:D     1 module hpm;
38:S
39:D     1 import
40:D     1   heapmax,      { integer/anyptr }
41:D     1   heapbase;    { integer/anyptr }
42:S
43:D     1 export
44:S
45:D     1 procedure Hestablish;
46:D     1 procedure New (var object:integer; bytesize:integer);
47:D     1 procedure Dispose (var object:integer; bytesize:integer);
48:D     1 procedure Mark (var nextfreeword:integer);
49:D     1 procedure Release (nextfreeword:integer);
50:S
51:D     1 implement
52:D     1 const
53:D     1   nilptr = 0;           {integer equivalent of nil pointer}
54:D     1   maxptr = 16777215;   {biggest physical address}
55:D     1   smallest = 2;       {smallest allocatable object (words)}
56:D     1   stdblk = 16;        {largest "standard" object}
57:D     1   otherlist = stdblk+1; {for nonstandard objects}
58:D     1 type
59:D     1   stdlists = smallest..stdblk;
60:D     1   allists = smallest..otherlist;

```

```

61:D     1 pointer = nilptr..maxptr;
62:D     1 pointertrick = packed record
63:D     1   case boolean of
64:D     1     true: { longform: integer };
65:D     1     false: { highbyte: 0..255;
66:D     1               lower3: pointer }
67:D     1   end;
68:D     1 mp = ^memblock;
69:D     1 memblock = packed record
70:D     1   smallsize: 0..255;
71:D     1   link: pointer;
72:D     1   case (smallsize=0) boolean of
73:D     1     true: { bigsize: integer }
74:D     1   end;
75:D     1 freearray = array [allists] of pointer;
76:D     1 dsap = ^dsa;
77:D     1 dsa = record
78:D     1   fakeone: memblock;
79:D     1   freelist: freearray;
80:D     1   wordsdisposed: integer;
81:D     1   dirty: boolean;
82:D     1 end;
83:S
84:S { procedure Newwords (var p: anyptr; wordlen: integer); external;
85:D     1
86:D     1 function longpointer (p: pointer): integer;
87:D     -4 2 var t: pointertrick;
88:D     2 begin
89:D     2   t.lower3 := p; t.highbyte := 255;
90:D     2   longpointer := t.longform
91:D     2 end;
92:S
93:D     1 function shortpointer (p: integer): pointer;
94:D     -4 2 var t: pointertrick;
95:D     2 begin
96:D     2   t.longform := p; shortpointer := t.lower3
97:D     2 end;
98:S
99:D     1 function getsize (p: pointer): integer;
100:D     -4 2 var size: integer;
101:D     2 begin
102:D     2   with mp(p)^ do
103:D     3   begin
104:D     3     size := mp(p)^.smallsize;
105:D     3     if size = 0 then size := bigsize;
106:D     3     getsize := size;
107:D     3   end;
108:D     2 end;
109:S
110:D     1 procedure putsize (p: pointer; size: integer);
111:D     2 begin
112:D     2   with mp(p)^ do
113:D     3   if size > otherlist then
114:D     4   begin smallsize := 0; bigsize := size end
115:D     4   else
116:D     4     smallsize := size;
117:D     2 end;
118:S
119:D     1 procedure Hestablish;
120:D     2 const

```



```
121:D      2      emptydsa :=
122:D      2      dsa [ fakeone:memblock[smallest-1,link:nilptr],
123:D      2      freelist:freearray[(otherlist-smallest+1) of nilptr],
124:D      2      wordsdisposed:0, dirty:false ];
125:D      2
126:D      -2      i: alllists;
127:C      2      begin
128:C      2      Newwords(anyptr(heapbase),(sizeof(dsa)+1) div 2);
129:C      2      dsap(heapbase)^ := emptydsa;
130:C      2      end; {Hestablisth}
131:S      1
132:D      1      procedure recombine;
133:D      2      label 1,2;
134:D      2      var
135:D      -28      fakeaddr,f,g,this,thislink,flimit,highmark: pointer;
136:D      -42      fsize,gsize,wordcount: integer; i: alllists;
137:C      2      begin
138:C      2      with dsap(heapbase)^ do
139:C      3      begin
140:C      3      highmark := shortpointer(heapmax);
141:C      3      fakeaddr := shortpointer(integer(addr(fakeone)));
142:C      3      fakeone.link := nilptr;
143:C      3      fakeone.smallsize := smallest-1;
144:C      3      for i:= smallest to otherlist do
145:C      4      while freelist[i] <> nilptr do
146:C      5      begin
147:C      5      i := freelist[i];
148:C      5      freelist[i] := mp(f)^.link;
149:C      5      this := fakeaddr;
150:C      5      thislink := mp(this)^.link;
151:C      5      while thislink <> nilptr do
152:C      6      if thislink > f then goto 2
153:C      7      else
154:C      7      begin
155:C      7      this := thislink;
156:C      7      thislink := mp(this)^.link;
157:C      7      end;
158:C      5      2: mp(f)^.link := thislink;
159:C      5      mp(this)^.link := f;
160:C      5      end;
161:C      3      f := fakeone.link; wordcount := 0;
162:C      3      while f <> nilptr do
163:C      4      begin
164:C      4      if highmark < f then goto 1;
165:C      4      fsize := getsize(f);
166:C      4      flimit := f+fsize+fsize;
167:C      4      while mp(f)^.link = flimit co (adjacent)
168:C      5      begin
169:C      5      g := mp(f)^.link;
170:C      5      mp(f)^.link := mp(g)^.link;
171:C      5      gsize := getsize(g);
172:C      5      flimit := g+gsize+gsize;
173:C      5      end; (while)
174:C      4      if highmark <= flimit then
175:C      4      begin
176:C      5      heapmax := longpointer(f);
177:C      5      goto 1;
178:C      4      end;
179:C      4      fsize := (flimit-f) div 2;
180:C      4      wordcount := wordcount+fsize;
```

```
181:C      4      putsze(f,fsize);
182:C      4      if fsize > otherlist then i:=otherlist
183:C      5      else i:=fsize;
184:C      4      g := mp(f)^.link;
185:C      4      mp(f)^.link := freelist[i];
186:C      4      freelist[i] := f;
187:C      4      f := g;
188:C      4      end; (f<>nil)
189:C      3      1: dirty := false; wordsdisposed := wordcount;
190:C      3      end; (with heapbase)
191:C      2      end; {recombine}
192:S      1
193:D      1      procedure hreturn (p:pointer; wordlen:integer);
194:D      -2      var list: alllists;
195:C      2      begin
196:C      2      with dsap(heapbase)^,mp(p)^ do
197:C      3      begin
198:C      3      if wordlen < 0
199:C      4      then wordlen := -wordlen
200:C      4      else wordsdisposed := wordsdisposed+wordlen;
201:C      3      dirty := true;
202:C      3      if wordlen > otherlist then list := otherlist
203:C      4      else list := wordlen;
204:C      3      putsze(p,wordlen);
205:C      3      link := freelist[list];
206:C      3      freelist[list] := p;
207:C      3      end; (with)
208:C      2      end; {hreturn}
209:S      1
210:D      1      procedure hallocate (var p:pointer; wordlen:integer);
211:D      -4      var longp: anyptr;
212:S      2
213:D      2      function tryit: pointer;
214:D      3      label 1,2;
215:D      3      var
216:D      -12      temp,prev,next: pointer;
217:D      -28      lfrag,j,len,size: integer;
218:C      3      begin
219:C      3      with dsap(heapbase)^ do
220:C      4      begin len := wordlen;
221:C      4      if len < otherlist then
222:C      5      begin
223:C      5      temp := freelist[len];
224:C      5      if temp <> nilptr then
225:C      6      begin
226:C      6      freelist[len] := mp(temp)^.link;
227:C      6      goto 1
228:C      6      end;
229:C      5      end;
230:C      4      lfrag := len+smallest;
231:C      4      for size := lfrag to stdblk do
232:C      5      begin
233:C      5      temp := freelist[size];
234:C      5      if temp <> nilptr then
235:C      6      begin
236:C      6      freelist[size] := mp(temp)^.link;
237:C      6      hreturn(temp+len+len, len+size);
238:C      6      goto 1
239:C      6      end;
240:C      5      end;
```



```

241:C      4      prev := nilptr; next := freelist[otherlist];
242:C      4      while next <> nilptr do
243:C      5      begin
244:C      5          size := getsize(next);
245:C      5          if (size=len) or (size>=lfrag) then goto 2;
246:C      5          prev := next; next := mp(next)^.link;
247:C      5      end;
248:C      4 2:      temp := next;
249:C      4      if next <> nilptr then
250:C      5      begin
251:C      5          if prev = nilptr
252:C      5              then freelist[otherlist] := mp(next)^.link
253:C      5              else mp(prev)^.link:=mp(next)^.link;
254:C      5          if size <> len then
255:C      5              hreturn(next+len+len,len-size);
256:C      5      end;
257:C      4 1:      if temp <> nilptr then
258:C      5          wordsdisposed := wordsdisposed+wordlen;
259:C      5          tryit := temp;
260:C      4      end; (with heapbase)
261:C      3      end; (tryit)
262:S
263:C      2      begin (hallocate)
264:C      2          p := tryit;
265:C      2          if (p=nilptr) and dsap(heapbase)^.dirty then
266:C      2              begin recombine; p := tryit end;
267:C      2          if p = nilptr then
268:C      3              begin
269:C      3                  Newwords(longp,wordlen);
270:C      3                  p := shortpointer(integer(longp));
271:C      3              end;
272:C      2      end; (hallocate)
273:S
274:D      1      procedure Mark (var nextfreeword:pointer);
275:C      2      begin
276:C      2          nextfreeword := heapmax;
277:C      2      end;
278:S
279:D      1      procedure Release (nextfreeword:integer);
280:C      2      begin
281:C      2          heapmax := nextfreeword;
282:C      2          if heapbase <> nilptr then recombine;
283:C      2      end;
284:S
285:D      -8 1      procedure New (var object:integer; bytesize:integer);
286:D      2      var ptr: pointer; wordsize: integer;
287:C      2      begin
288:C      2          wordsize := (bytesize+1) div 2;
289:C      2          if heapbase = nilptr then
290:C      3              Newwords(anyptr(object),wordsize)
291:C      3          else
292:C      3              begin
293:C      3                  if wordsize < smallest then wordsize := smallest;
294:C      3                  if dsap(heapbase)^.wordsdisposed < wordsize then
295:C      4                      Newwords(anyptr(object),wordsize)
296:C      4                  else
297:C      4                      begin
298:C      4                          hallocate(ptr,wordsize);
299:C      4                          object := longpointer(ptr);
300:C      4                      end;

```

```

301:C      3      end;
302:C      2      end; (New)
303:S
304:D      -8 1      procedure Dispose (var object:integer; bytesize:integer);
305:D      2      var ptr: pointer; wordsize: integer;
306:C      2      begin
307:C      2          if heapbase <> nilptr then
308:C      3              begin
309:C      3                  wordsize := (bytesize+1) div 2;
310:C      3                  if wordsize < smallest then wordsize := smallest;
311:C      3                  ptr:=shortpointer(object);
312:C      3                  if ptr = nilptr then escape(-3);
313:C      3                  hreturn(ptr,wordsize)
314:C      3              end;
315:C      2          object := nilptr;
316:C      2      end; (Dispose)
317:S
318:C      1 end. (heapmanager)
319:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

HPHIL

Description

HPHIL provides driver routines for the Hewlett-Packard Human Interface Link (HP-HIL).

Requirements

A804XDVR, SYSGLOBALS, SYSDEVS, ISR, and ASM.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1984.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S     RESTRICTED RIGHTS LEGEND
11:S
12:S     Use, duplication, or disclosure by the Government
13:S     is subject to restrictions as set forth in
14:S     paragraph (b) (3) (B) of the Rights in Technical
15:S     Data and Computer Software clause in
16:S     DAR 7-104.9(a).
17:S
18:S     HEWLETT-PACKARD COMPANY
19:D     0 Fort Collins, Colorado      *)
20:S
21:D     0 $sysprog$
22:D     0 $stackcheck off$ $heap dispose off$
23:D     0 $iocheck off$ $range off$ $ovflcheck off$
24:D     0 $debug off$
25:S
26:D     0 program loopinit;
27:S
28:D     1 module hphil; { HP Human Interface Loop }
29:S
30:D     1 import a804xdvr,sysdevs,sysglobals,asm;
31:S
32:D     1 export
33:D     1 FUNCTION INITLOOP:BOOLEAN;      (4/6/84 SFB)
34:S
35:D     1 implement
36:D     1 const
37:D     1   readctrl = hex('FB'); { 8042 COMMAND5 }
38:D     1   writectrl = hex('EB');
39:D     1   startcmd = hex('E0');
40:D     1   readbusy = hex('02');
41:D     1   trigger = hex('C5');
42:S
43:S
44:D     1 type
45:D     1   stat5datatype = packed record { data byte with status 5 }
46:D     1   case integer of
47:D     1     0 : (b:byte);
48:D     1     1 : (lerror : boolean;          {7}
49:D     1         padding : 0..3;             {6..5}
50:D     1         polling : boolean;          {4}
51:D     1         command : boolean;          {3}
52:D     1         devaddr : 0..7;            {2..0}
53:D     1   end;
54:S
55:D     1   lpcctrltype = packed record { loop control byte }
56:D     1   case boolean of
57:D     1     true : (b:byte);
58:D     1     false : (doreconfigure: boolean;
59:D     1           pad6,pad5 : boolean;
60:D     1           cookkbd : boolean;

```

```

61:D     1   pad3 : boolean;
62:D     1   noreconfig : boolean;
63:D     1   noerts : boolean;
64:D     1   dosutopoll : boolean )
65:D     1 end;
66:S
67:D     1 var
68:D     -8 1 oldstatus5lsr : kbdhooktype;
69:D     -16 1 oldclrio : procedure;
70:S
71:D     1 procedure dummydataproc(var statbyte,databyte:byte; var done:boolean);
72:C     2 begin end;
73:D     1 procedure dummyopsproc(op:loopdvrop);
74:C     2 begin end;
75:S
76:D     1 procedure lcommand;
77:D     2 const
78:D     2   timeout20 = hex('FE');
79:D     2 begin
80:C     3 WITH LOOPCONTROL^ DO      (4/6/84 SFB)
81:C     3   if loopisok or loopinconfig then
82:C     4   begin
83:C     4     sendcmd(startcmd);
84:C     4     senddata(8*loopdevice); { command & address }
85:C     4     senddata(loopcmd);      { id command }
86:C     4     senddata(timeout20);    { 20 mill secs }
87:C     4     loopcmddone:=false; looperror:=false;
88:C     4     sendcmd(trigger);
89:C     4     repeat call(kbdpollhook,true);
90:C     4     until loopcmddone or looperror;
91:C     4     end
92:C     4   else looperror:=true;
93:C     2 end; { lcommand }
94:S
95:D     -16 1 { status6 lsr for describe operation }
96:D     1 procedure describedata(var statbyte,databyte:byte; var done:boolean);
97:C     2 begin { load describe record for the current loop device }
98:C     2 WITH LOOPCONTROL^ DO      (4/10/84 SFB)
99:C     3   with loopdevices[loopdevice] do {4/6/84 SFB}
100:C     4   case devstate of
101:C     5     1: begin descrip.darray[1]:=chr(databyte); devstate:=2; end;
102:C     5     2: begin
103:C     5       descrip.darray[2]:=chr(databyte); { header byte }
104:C     5       if descrip.numaxes>0 then devstate:=3
105:C     5       else devstate:=11;
106:C     5     end;
107:C     5     3: begin descrip.darray[4]:=chr(databyte); devstate:=4; end;
108:C     5     4: begin
109:C     5       descrip.darray[3]:=chr(databyte); { high order count }
110:C     5       if descrip.abscoords then devstate:=5
111:C     5       else devstate:=11;
112:C     5     end;
113:C     5     5..10: begin { swap even / odd bytes }
114:C     5       descrip.darray[devstate-1+(ord(odd(devstate))*2)]:=chr(databyte);
115:C     5       devstate:=devstate+ord(not devstate<16);
116:C     5       if (devstate>6) and (descrip.numaxes=1) then devstate:=11
117:C     5       else
118:C     5         if (devstate>8) and (descrip.numaxes=2) then devstate:=11;
119:C     5       end;
120:C     5   11: begin { prompts & buttons }

```



```

121:C      5      if descrip.hasprompts then descrip.darray[11]:=chr(databyte);
122:C      5      devstate:=12;
123:C      5      end;
124:C      5      otherwise
125:C      5      end; { case devstate }
126:C      2      end; { describedata }
127:S
128:S
129:D      1      procedure rawshift; { change 8042 to raw mode }
130:D      2      var
131:D      -4      2      oldpriority : integer;
132:D      -6      2      ctrlreg   : lpctrltype;
133:D      -8      2      tempstuff  : byte;
134:C      2      begin
135:C      2      WITH LOOPCONTROL^ DO {4/6/84 SF8}
136:C      3      if not rawmode then
137:C      4      begin
138:C      4      oldpriority:=intlevel; { lock out interrupts }
139:C      4      setintlevel(7);
140:C      4      cmd_read_1(readctrl,tempstuff); ctrlreg.b:=tempstuff;
141:S
142:C      4      ctrlreg.doautopoll:=false;
143:C      4      ctrlreg.noerrs:=true;
144:C      4      ctrlreg.noreconf:=false;
145:C      4      sendcmd(writectrl);
146:C      4      senddata(ord(ctrlreg.b));
147:S
148:C      4      repeat { wait for polling to stop }
149:C      5      cmd_read_1(readbusy,loopdata);
150:C      5      until not Odd(loopdata div 4);
151:C      4      rawmode:=true;
152:C      4      setintlevel(oldpriority); { restore interrupts }
153:C      4      end;
154:C      2      end; { rawshift }
155:S
156:D      1      procedure normshift; { change 8042 to normal mode }
157:D      2      var
158:D      -4      2      oldpriority : integer;
159:D      -6      2      ctrlreg   : lpctrltype;
160:D      -8      2      tempstuff  : byte;
161:C      2      begin
162:C      2      WITH LOOPCONTROL^ DO {4/6/84 SF8}
163:C      3      if rawmode then
164:C      4      begin
165:C      4      oldpriority:=intlevel; { lock out interrupts }
166:C      4      setintlevel(7);
167:C      4      cmd_read_1(readctrl,tempstuff); ctrlreg.b:=tempstuff;
168:C      4      ctrlreg.doautopoll:=true;
169:C      4      ctrlreg.noerrs:=true;
170:C      4      ctrlreg.noreconf:=false;
171:S
172:C      4      sendcmd(writectrl); senddata(ord(ctrlreg.b)); { start auto polling }
173:S
174:C      4      rawmode := false;
175:C      4      setintlevel(oldpriority);
176:C      4      end;
177:C      2      end; { normshift }
178:S
179:D      1      procedure describe; { issue describe for loopdevice }
180:D      2      const

```

```

181:D      2      idcmd   = hex('03');
182:S
183:D      2      var
184:D      -4      2      oldpriority : integer;
185:D      -6      2      i           : 1..11;
186:C      2      begin { describe }
187:C      2      oldpriority := intlevel; { lockout interrupts }
188:C      2      setintlevel(7);
189:C      2      statushook:=describedata;
190:C      2      WITH LOOPCONTROL^ DO {4/10/84 SF8}
191:C      3      with loopdevices[loopdevice] do
192:C      4      begin
193:C      4      devstate:=1;
194:C      4      opsproc:=dummyopsproc;
195:C      4      dataproc:=dummydataproc;
196:C      4      for i:=1 to 11 do descrip.darray[i]:=chr(0); { clear the record }
197:C      4      loopcmd:=idcmd;
198:C      4      lcomand; { send the command and wait for it to finish }
199:C      4      end;
200:C      2      statushook:=dummydataproc;
201:C      2      setintlevel(oldpriority);
202:C      2      end; { describe }
203:S
204:D      1      procedure checkloop;
205:D      2      const
206:D      2      readlstat = hex('FA');
207:D      2      var
208:D      -4      2      oldpriority : integer;
209:C      2      begin
210:C      2      WITH LOOPCONTROL^ DO {4/6/84 SF8}
211:C      3      BEGIN
212:C      3      loopisok:=false;
213:C      3      LOOPDATA := 0; {4/24/84 SF8 }
214:C      3      oldpriority:=intlevel; { lock out interrupts }
215:C      3      setintlevel(7);
216:C      3      cmd_read_1(readlstat,loopdata);
217:C      3      loopisok:=loopdata<128;
218:C      3      setintlevel(oldpriority);
219:C      3      END; {4/6/84 SF8}
220:C      2      end; { checkloop }
221:S
222:D      1      procedure configure;
223:D      2      label 1;
224:D      2      CONST
225:D      2      READ_INTR_MASK = 4; { SF8 3/5/84 }
226:D      2      var
227:D      -4      2      temp : loopdvprtr;
228:D      -6      2      i : shortint;
229:D      -8      2      OLDMASK : 8YTE; { SF8 3/5/84 }
230:C      2      begin { configure }
231:C      2      WITH LOOPCONTROL^ DO {4/6/84 SF8}
232:C      3      BEGIN
233:C      3      CMO_READ_1(READ_INTR_MASK,OLDMASK); { SAVE OLD 804X MASK - SF8 3/5/84 }
234:C      3      CALL(MASKOPSHOOK,K8DMASK,0); { ENABLE LOOP - SF8 3/5/84 }
235:C      3      loopdevreading:=false; { 3.0 BUG #39 3/17/84 }
236:C      3      checkloop;
237:C      3      if not loopisok then goto 1;
238:C      3      rawshift;
239:C      3      loopinconfig:=true;
240:C      3      { get device description for all (7) devices }

```



```

241:C      3   for i:=1 to 7 do
242:C      4   begin
243:C      4   loopdevice:=i;
244:C      4   describe;
245:C      4   if looperror then goto 1; { kick out & wait for status5, le_configured }
246:C      4   end;
247:S
248:C      3   { find and attach drivers to devices }
249:C      3   for i:=1 to 7 do
250:C      4   begin
251:C      4   loopdevice:=i;
252:C      4   temp:=loopdriverlist;
253:C      4   while temp<>nil do
254:C      5   with loopdevices[loopdevice] do
255:C      6   begin
256:C      6   if (temp^.daddr=0) or (temp^.daddr=loopdevice) then
257:C      7   if (descrip.id=temp^.lowid) and (descrip.id<=temp^.highid) then
258:C      8   begin { hook up the driver }
259:C      8   opsproc:=temp^.opsproc;
260:C      8   dataproc:=temp^.dataproc;
261:C      8   call(opsproc,resetdevice); { tell driver to reset }
262:C      8   temp:=nil; { cancel the search }
263:C      8   end
264:C      8   else temp:=temp^.next
265:C      8   else temp:=temp^.next; { 3.0 BUG FIX 4/6/84 }
266:C      6   end; { while .. with }
267:C      4   end; { for }
268:S
269:C      3   1:loopinconfig:=false;
270:C      3   normshift;
271:C      3   IF ODD(OLDMASK) THEN
272:C      3   CALL(MASKOPSHOOK,0,KBDMASK); { DISABLE LOOP IF IT WAS PREVIOUSLY
273:C      3   OFF - SFB 3/5/84 }
274:C      3   END; { 4/6/84 SFB }
275:C      2   end; { configure }
276:S
277:D      1   procedure resettheloo; { cleario call }
278:C      2   begin
279:C      2   configure; call(oldclrrio);
280:C      2   end;
281:S
282:D      1   procedure endcmdproc(var statbyte,databyte:byte; var done:boolean);
283:C      2   begin
284:C      2   WITH LOOPCONTROL^ DO { 4/6/84 SFB }
285:C      3   BEGIN
286:C      3   loopcmddone := true; status6hook:=dummydataproc;
287:C      3   loopdata := databyte;
288:C      3   END;
289:C      2   end;
290:S
291:D      1   PROCEDURE SENDHPHILCMD(OP : HPHILOP); { 4/9/84 SFB }
292:C      2   BEGIN
293:C      2   CASE OP OF
294:C      3   RAWSHIFTOP : RAWSHIFT;
295:C      3   NORMSHIFTOP : NORMSHIFT;
296:C      3   CHECKLOOPOP : CHECKLOOP;
297:C      3   CONFIGUREOP : CONFIGURE;
298:C      3   LCOMMANDOP : LCOMMAND;
299:C      3   END;
300:C      2   END;

```

```

301:S
302:D      1   procedure status5isr(var statbyte,databyte:byte; var done:boolean);
303:C      2   var status : stat5datatype;
304:C      2   begin
305:C      3   status.b:=databyte;
306:C      2   WITH LOOPCONTROL^ DO { 4/6/84 SFB }
307:C      3   if status.lerror then
308:C      4   begin
309:C      4   looperror:=true;
310:C      4   status6hook:=dummydataproc;
311:C      4   if status.b=le_loopdown then
312:C      5   loopisok:=false
313:C      5   else
314:C      5   if status.b=le_configured then
315:C      6   begin { loop has been reconfigured }
316:C      6   call(oldstatus5isr,statbyte,databyte,done); { let keyboard proc know }
317:C      6   if not loopinconfig then configure;
318:C      6   loopdevreading:=false; { 3.0 bug #39 3/17/84 }
319:C      5   end;
320:C      4   end
321:C      3   else
322:C      4   if status.command then
323:C      5   begin
324:C      5   if status.polling then { end of polled data }
325:C      6   begin
326:C      6   status6hook:=dummydataproc; { ignore next data }
327:C      6   call(loopdevices[loopdevice].opsproc,dataended); { 3.0 bug #39 3/17/84 }
328:C      6   loopdevreading:=false; { 3.0 bug #39 3/17/84 }
329:C      5   end
330:C      5   else
331:C      6   begin { command ending }
332:C      6   status6hook:=endcmdproc;
333:C      6   end;
334:C      5   end
335:C      3   else
336:C      5   if status.polling then
337:C      6   begin { polling data starting }
338:C      6   { check if device now reading 3.0 bug #39 3/17/84 }
339:C      6   if loopdevreading then { 3.0 bug #39 3/17/84 }
340:C      7   call(loopdevices[loopdevice].opsproc, dataended); { 3.0 bug #39 3/17/84 }
341:C      7   else { 3.0 bug #39 3/17/84 }
342:C      6   loopdevreading:=true;
343:C      6   loopdevice:=status.devaddr;
344:C      6   status6hook:=loopdevices[loopdevice].dataproc;
345:C      6   call(loopdevices[loopdevice].opsproc,datastarting);
346:C      6   end
347:C      5   else
348:C      6   begin { data starting for current command }
349:C      6   looperror:= loopdevice<>status.devaddr;
350:C      6   end;
351:C      3   end; { status5isr }
352:S
353:D      1   function initloop:boolean;
354:C      2   begin
355:C      2   IF LOOPCONTROL = NIL THEN { 4/9/84 SFB }
356:C      3   BEGIN
357:C      3   NEW(LOOPCONTROL); { 4/9/84 SFB }
358:C      3   WITH LOOPCONTROL^ DO { 4/6/84 SFB }
359:C      4   BEGIN
360:C      4   initloop:=false;

```



```

361:C      4      rawmode :=false;
362:C      4      loopisok :=false;
363:C
364:C      4      if kbdttype=itfkbd then
365:C          5      begin
366:C              5      oldstatusSisr:=statusShook;      ( save old hook )
367:C              5      statusShook:=statusSisr;
368:C              5      statusShook:=dummydataproc;
369:C              5      oldclrio :=cleariohook;      ( save old hook )
370:C              5      cleariohook:=resettheloo;
371:C              5      HPHILCMDHOOK:=SENHHPHILCMD;      (4/9/84 SFB)
372:C              5      configure;
373:C              5      initloop := true;
374:C              5      end;
375:C          4      END;      (WITH LOOPCONTROL^)
376:C          3      END      (IF LOOPCONTROL = NIL)
377:C      3      ELSE
378:C          3      BEGIN      (4/9/84 SFB)
379:C              3      CONFIGURE;      (IF INITLOOP ALREADY DONE JUST CONFIGURE)
380:C              3      INITLOOP := FALSE;      (DON'T MARKUSER)
381:C              3      END;
382:C          2      end; { initloop }
383:C
384:C      1 end; { loopinterface module }
385:C
386:C      1 import hphil, loader;
387:C
388:C      1 begin
389:C          1 if initloop then markuser;
390:C          1 end.
391:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

INIT

Description

INIT contains the following:

- **ISR:** Installs and removes interrupt service handlers using ISRIBs.
- **MISC:** Miscellaneous support utilities and DAM for unblocked devices.
- **FS:** File support routines. The IMPLEMENT portion of the listing has been suppressed due to the proprietary nature of the software.
- **INITUNITS:** Creates the unit table.
- **LDR:** Interface to Loader and miscellaneous support.
- **SETUPSYS:** The initialization.

Usage

INIT is used by the operating system (OS) kernel (in INITLIB).

Many routines in FS are called directly by compiler generated code.

Notes

Includes: ISR, MISC, FS, INITUNITS, LDR and SETUPSYS.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```
1: (*
2:
3: (c) Copyright Hewlett-Packard Company, 1983.
4: All rights are reserved. Copying or other
5: reproduction of this program except for archival
6: purposes is prohibited without the prior
7: written consent of Hewlett-Packard Company.
8:
9:
10: RESTRICTED RIGHTS LEGEND
11:
12: Use, duplication, or disclosure by the Government
13: is subject to restrictions as set forth in
14: paragraph (b) (3) (B) of the Rights in Technical
15: Data and Computer Software clause in
16: DAR 7-104.9(a).
17:
18: HEWLETT-PACKARD COMPANY
19: 0 Fort Collins, Colorado *)
20:
21:
22: 0 $modcal$
23: 0 $heap_dispose off$
24: 0 $iocheck off$
25: 0 $range off$ $ovflcheck off$
26: 0 $debug off$
27: 0 $stackcheck off$
28: 0 $search 'INITLOAD', 'ASM'$
29:
30: 0 program setupsys(input, output, keyboard, listing);
31:
32: 1 $include 'ISR'$;
33: 1 module isr;
34:
35: 1 import sysglobals,asm;
36:
37: 1 export
38:
39: 1 procedure isrlink(procentry : isrproctype;
40: 2 lintregaddr : charptr;
41: 2 lintregmask : byte;
42: 2 lintregvalue : byte;
43: 2 lintlevel : byte;
44: 2 isribp : pisrib);
45:
46: 1 procedure permisrlink(procentry : isrproctype;
47: 2 lintregaddr : charptr;
48: 2 lintregmask : byte;
49: 2 lintregvalue : byte;
50: 2 lintlevel : byte;
51: 2 isribp : pisrib);
52:
53: 1 procedure isrunlink(lintlevel : byte;
54: 2 isribp : pisrib);
55:
56: 1 procedure isrchange(procentry : isrproctype;
57: 2 isribp : pisrib);
58:
59: 1 implement
```

```
60: 1 procedure isrlink(procentry : isrproctype;
61: 2 lintregaddr : charptr;
62: 2 lintregmask : byte;
63: 2 lintregvalue : byte;
64: 2 lintlevel : byte;
65: 2 isribp : pisrib);
66: 2 begin
67: 2 with isribp^ do
68: 2 begin
69: 2 lintregaddr := lintregaddr;
70: 2 lintregmask := lintregmask;
71: 2 lintregvalue := lintregvalue;
72: 2 proc := procentry;
73: 2 link := interrupttable[lintlevel];
74: 2 end;
75: 2 interrupttable[lintlevel]:=isribp;
76: 2 end;
77:
78: 1 procedure permisrlink(procentry : isrproctype;
79: 2 lintregaddr : charptr;
80: 2 lintregmask : byte;
81: 2 lintregvalue : byte;
82: 2 lintlevel : byte;
83: 2 isribp : pisrib);
84: 2 var
85: 2 -4 nxltnk : pisrib;
86: 2 -8 plinkp : ^pisrib;
87:
88: 2 begin
89: 2 nxltnk := interrupttable[lintlevel];
90: 2 plinkp := addr(interrupttable[lintlevel]);
91:
92: 2 while nxltnk <> perminttable[lintlevel] do
93: 2 begin
94: 2 plinkp := addr(nxltnk^.link);
95: 2 nxltnk := nxltnk^.link;
96: 2 end;
97:
98: 2 with isribp^ do
99: 2 begin
100: 2 lintregaddr := lintregaddr;
101: 2 lintregmask := lintregmask;
102: 2 lintregvalue := lintregvalue;
103: 2 proc := procentry;
104: 2 link := nxltnk;
105: 2 end;
106:
107: 2 plinkp^ := isribp;
108: 2 perminttable[lintlevel] := isribp;
109: 2 end;
110:
111: 1 procedure isrunlink(lintlevel : byte;
112: 2 isribp : pisrib);
113: 2 var
114: 2 -4 nxltnk : pisrib;
115: 2 -8 plinkp : ^pisrib;
116:
117: 2 begin
118: 2 nxltnk := interrupttable[lintlevel];
119: 2 plinkp := addr(interrupttable[lintlevel]);
```



```

120:S      2   while (nxtlink <> nil) and (nxtlink <> isribp) do
121:C      3       begin
122:C      4           plinkp := addr(nxtlink^.link);
123:C      5           nxtlink := nxtlink^.link;
124:C      6       end;
125:C
126:S      2   if nxtlink <> nil then
127:C      3       begin
128:C      4           if perminttable[lintlevel] = nxtlink then
129:C      5               perminttable[lintlevel] := nxtlink^.link;
130:C      6           plinkp^ := nxtlink^.link;
131:C      7       end;
132:C
133:C      2 end;
134:C
135:S      1 procedure isrchange(procentry : isrproctype;
136:D      2                     isribp : pisrib);
137:D      3 var
138:D      4     saveintlevel:integer;
139:D      5 -4
140:S      6 2 begin
141:C      7     saveintlevel:=intlevel;
142:C      8     setintlevel(7); (disable all interrupts)
143:C      9     with isribp^ do proc:=procentry;
144:C     10     setintlevel(saveintlevel);
145:C     11 end;
146:C
147:S      1 end {module isr}
148:C
149:S
150:C      2 $include 'ISR'$.
150:D      1 $include 'MISC'$.
151:D      1 module misc; {homeless orphans}
152:S
153:D      1 import sysglobals, asm;
154:S
155:D      1 export
156:S
157:D      1 const
158:S
159:D      1     null = 0;     nulldchar = chr(0);
160:D      2                     homechar = chr(1);
161:D
162:D      1     etx = 3;
163:D      2                     bellchar = chr(7);
164:D      3                     bs = 8;     leftchar = chr(8);
165:D      4                     tab = 9;    cteol = chr(9);
166:D      5                     lf = 10;   downchar = chr(10);
167:D      6                     vt = 11;   cteos = chr(11);
168:D      7                     ff = 12;   clearscr = chr(12);
169:D      8                     cr = 13;   eol = chr(13);
170:D      9                     dle = 16;
171:D     10                     esc = 27;   escchar = chr(27);
172:D     11                     fsp = 28;  rightchar = chr(28);
173:D     12                     us = 31;  upchar = chr(31);
174:D     13                     del = 127;
175:D     14                     cntrl = 255;
176:S
177:D      1 type
178:S      1 (* CATALOGUE INFORMATION, zero entry refers to the directory itself *)

```

```

179:D      1 catentry = record
180:D      2     cname: tid;           {name of file or directory}
181:D      3     cft: shortint;        {external file type (LIF)}
182:D      4     ckind: filekind;      {file kind}
183:D      5     cpsize: integer;      {physical size of file
184:D      6                             or of total data space on volume}
185:D      7     clsize: integer;      {logical size of file
186:D      8                             or unused space on medium}
187:D      9     cstart: integer;     {starting location of file
188:D     10                             or first possible data location}
189:D     11     cblocksize: integer; {size of a sector or block}
190:D     12     createdate, clastdate: daterec; {creation, last modified dates}
191:D     13     ccreateime, clasttime: timerec; {creation, last modified times}
192:D     14     cextra1: integer;     {extension
193:D     15                             or total possible number of files
194:D     16                             or requested number of files}
195:D     17     cextra2: integer;     {secondary discretionary field
196:D     18                             or start index of requested catalog}
197:D     19     cinfo: string[20];    {comment or miscellaneous information}
198:D
199:D      1 end;
200:S
201:D      1 passentry = record
202:D      2     pbits: integer;
203:D      3     pword: passtype;
204:D      4     end;
205:S
206:D      1 var
207:D      2 -2 1 idle: byte;           { idle character -- 3.0 bug jws 3/20/84 }
208:S
209:D      2 -2 1 procedure getioerrmsg(var s:string; lastior: integer);
210:D      3 -2 1 procedure printerror(errorcode, lastior: integer);
211:D      4 -2 1 procedure upc(var s: string);
212:D      5 -2 1 function ueovbytes(unit: unitnum): integer;
213:D      6 -2 1 procedure unlockddam(anyvar f: fib; unum: unitnum; request: damrequesttype);
214:D      7 -2 1 procedure initfilekinds;
215:D      8 -2 1 procedure lockup;
216:D      9 -2 1 procedure lockdown;
217:S
218:S
219:D      2 -2 1 implement
220:S
221:D      3 1 procedure lockup;
222:C      4 2 begin
223:C      5 3     locklevel := locklevel + 1;
224:C      6 4 end;
225:S
226:D      7 1 procedure lockdown;
227:D      8 -4 var saveio: integer;
228:D      9 -6 saveec: shortint;
229:C     10 2 begin
230:C     11 3     locklevel := locklevel - 1;
231:C     12 4 if locklevel = 0 then
232:C     13 5     begin
233:C     14 6         saveio := ioresult;
234:C     15 7         saveec := sysescapcode;
235:C     16 8         while actionspending > 0 do
236:C     17 9             begin
237:C     18 10                 call[deferredaction[actionspending]];
238:C     19 11                 actionspending := actionspending - 1;

```



```

239:C      4      end;
240:C      3      ioresult := saveio;
241:C      3      sysescapecode := saveec;
242:C      3      end;
243:C      2      end;
244:C
245:C
246:D      1      procedure upc(var s: string);
247:D      -2      var i: shortint;
248:D      -3      c: char;
249:C      2      begin
250:C      2      for i := 1 to strlen(s) do
251:C      3      begin
252:C      3      c := s[i];      (this assignment saves two or three subscripts)
253:C      3      if (c >= 'a') and (c <= 'z') then s[i] := chr(ord(c)+ (ord('A')-ord('a')));
254:C      3      end;
255:C      2      end;
256:C
257:D      1      function ueovbytes(unit: unitnum): integer;
258:D      2      label 1;
259:D      -4      var return_bytes: integer;
260:D      -8      saved_ioresult: integer;
261:D      -9      saved_reportbit: boolean;
262:D      -12      tracks: shortint;
263:D      -674      f: fib;
264:C
265:D      2      function try_tracks(number_of_tracks: shortint): boolean;
266:C      3      begin (try_tracks)
267:C      3      f.fpeof := number_of_tracks*(30*256)-unitable[unit].byteoffset;
268:C      3      f.funit := unit; f.fileid := 0;
269:C      3      call (unitable[unit].tm, addr(f), readbytes, f.fbuffer, 256, f.fpeof-512);
270:C      3      if ioresult=0 then
271:C      4      begin
272:C      4      return_bytes := f.fpeof; ( <<< desired side-effect! )
273:C      4      try_tracks := true
274:C      4      end (if)
275:C      4      else
276:C      4      try_tracks := false
277:C      4      end; (try_tracks)
278:C
279:C      2      begin (ueovbytes)
280:C      2      with unitable[unit] do
281:C      3      if not usbikd then ueovbytes := maxint
282:C      3      else if (letter<>'H') and (letter<>'F') then ueovbytes := umaxbytes
283:C      3      else
284:C      4      begin
285:C      4      saved_reportbit := ureportchange; ureportchange := false;
286:C      4      saved_ioresult := ioresult;
287:C      4      return_bytes := 0;      (in case none of the tries succeed!!!)
288:C      4      if letter='H' then
289:C      5      if try_tracks(150) then goto 1;      (D5 disc init'ed by a 9895)
290:C      5      for tracks := 61 to 67 do
291:C      5      if not try_tracks(tracks) then goto 1;      (55 disc init'ed by a 9895)
292:C      5      if not try_tracks(73) then (goto 1);      (55 disc init'ed by a 9895)
293:C      5      1: ioresult := saved_ioresult;
294:C      5      ureportchange := saved_reportbit;
295:C      5      if return_bytes<=umaxbytes then ueovbytes := return_bytes
296:C      5      else ueovbytes := umaxbytes;
297:C      4      end;
298:C      2      end; (ueovbytes)

```

```

299:C
300:D      -4      1      procedure getioerrmsg(var s: string; lastior: integer);
301:D      2      var dummy: integer;
302:C      2      begin
303:C      2      case lastior of
304:C      3      ord(iioerror)      : s := '(no I/O error reported)';
305:C      3      ord(zbadblock)    : s := 'block parity error';
306:C      3      ord(ibadunit)     : s := 'illegal unit number';
307:C      3      ord(zbadmode)     : s := 'illegal I/O request';
308:C      3      ord(ztimeout)    : s := 'device timeout';
309:C      3      ord(ilostunit)    : s := 'volume has gone off-line';
310:C      3      ord(ilostfile)   : s := 'file lost in dir';
311:C      3      ord(ibadtitle)   : s := 'bad file name';
312:C      3      ord(inoroom)     : s := 'no room on vol';
313:C      3      ord(inounit)     : s := 'logical volume not found';
314:C      3      ord(inofile)     : s := 'file not found';
315:C      3      ord(idupfile)    : s := 'dup dir entry';
316:C      3      ord(inotclosed) : s := 'file already open';
317:C      3      ord(inotopen)    : s := 'file not open';
318:C      3      ord(ibadformat)  : s := 'bad input format';
319:C      3      ord(znosuchblk)   : s := 'block number out of range';
320:C      3      ord(znodevice)   : s := 'device absent or unaccessible';
321:C      3      ord(zinitfail)   : s := 'medium formatting/sparing failed';
322:C      3      ord(zprotected) : s := 'medium is write protected';
323:C      3      ord(zstrange)   : s := 'unexpected interrupt';
324:C      3      ord(zbadhardware) : s := 'hardware fault';
325:C      3      ord(zcatchall)   : s := 'unrecognized error state';
326:C      3      ord(zbaddma)     : s := 'DMA absent or unavailable';
327:C      3      ord(inotvalidsize) : s := 'file size not compatible with type';
328:C      3      ord(inotreadable) : s := 'file not opened for reading';
329:C      3      ord(inotwriteable) : s := 'file not opened for writing';
330:C      3      ord(inotdirect)  : s := 'file not opened for direct access';
331:C      3      ord(idirfull)    : s := 'no room in directory';
332:C      3      ord(istrovfl)    : s := 'string subscript out of range';
333:C      3      ord(ibadclos)    : s := 'bad file close string parameter';
334:C      3      ord(ioeof)       : s := 'tried to read or write past eof';
335:C      3      ord(zuninitialized) : s := 'medium uninitialized';
336:C      3      ord(znblock)     : s := 'block not found';
337:C      3      ord(znotready)   : s := 'device not ready or medium absent';
338:C      3      ord(znomedium)   : s := 'medium absent';
339:C      3      ord(inodirectory) : s := 'no directory on volume';
340:C      3      ord(ibadfiletype) : s := 'file type illegal or does not match';
341:C      3      ord(ibadvalue)    : s := 'parameter illegal or out of range';
342:C      3      ord(icanststretch) : s := 'file cannot be extended';
343:C      3      ord(ibadrequest) : s := 'undefined operation for unit/file';
344:C      3      ord(inotlockable) : s := 'file not lockable';
345:C      3      ord(ifilealreadylocked) : s := 'file already locked';
346:C      3      ord(ifileunlocked) : s := 'file not locked';
347:C      3      ord(idirnotempty) : s := 'directory not empty';
348:C      3      ord(itoomanyopen) : s := 'too many files open on device';
349:C      3      ord(inoaccess)    : s := 'access to file not allowed';
350:C      3      ord(ibadpass)     : s := 'invalid password';
351:C      3      ord(ifilenotdir)  : s := 'file is not a directory';
352:C      3      ord(inotondir)   : s := 'operation not allowed on directory';
353:C      3      ord(ineedtempdir) : s := 'cannot create /WORKSTATIONS/TEMP_FILES';
354:C      3      ord(isrmcatchall) : s := 'unrecognized SRM error';
355:C      3      ord(zmediumchanged) : s := 'medium may have been changed';
356:C
357:C      3      otherwise
358:C      3      begin setstrlen(s,0);

```



```

359:C      3      strwrite(s, 1, dummy, 'ioresult was ', lastior:1);
360:C      3      end;
361:C
362:C      3      end (*IO ERRORS*);
363:C      2      end;
364:C
365:D      1      procedure printerror(errorcode,lastior: integer);
366:D      2      label 1;
367:D      2      var s,st: string[80];
368:D      2      -164      excp_line['excp_line']: integer;
369:D      2      -164      excp_pc['excp_pc']: integer;
370:C      2      begin
371:C      2      writeln(cteol,bellchar); write(cteol);
372:C
373:C      2      if errorcode > 0 then
374:C      3      writeln(output,'Abnormal termination. Halt code ',errorcode:1)
375:C      3      else
376:C      3      begin
377:C      4      case errorcode of
378:C      4      0: s := 'normal termination not caught by G0';
379:C      4      -1: s := 'abnormal termination not caught by G0';
380:C      4      -2: s := 'not enough memory';
381:C      4      -3: s := 'reference to NIL pointer';
382:C      4      -4: s := 'integer overflow';
383:C      4      -5: s := 'divide by zero';
384:C      4      -6: s := 'real math overflow';
385:C      4      -7: s := 'real math underflow';
386:C      4      -8: s := 'value range error';
387:C      4      -9: s := 'case value range error';
388:C
389:C      4      -10: getioerrmsg(s,lastior);      (* IORESULT <> 0 *)
390:C
391:C      4      -11: s := 'CPU word access to odd address';
392:C      4      -12: s := 'CPU bus error';
393:C      4      -13: s := 'illegal CPU instruction';
394:C      4      -14: s := 'CPU privilege violation';
395:C      4      -15: s := 'bad argument: SIN/COS';
396:C      4      -16: s := 'bad argument: LN';
397:C      4      -17: s := 'bad argument: SQR';
398:C      4      -18: s := 'bad argument: real/BCD conversion';
399:C      4      -19: s := 'bad argument: BCD/real conversion';
400:C      4      -20: s := 'stopped by user';
401:C      4      -21: s := 'unassigned CPU trap';
402:C      4      (*-22: s := '***** call to debugger *****'*)
403:C      4      -23: goto 1;      (****** give no message !! ******)
404:C      4      -24: s := 'macro parameter not 0..9 or A..Z';
405:C      4      -25: s := 'undefined macro parameter';
406:C      4      -26: s := 'I/O routine error';
407:C      4      -27: s := 'graphics routine error';
408:C      4      -28: s := 'ram parity error';
409:C      4      -29: s := 'misc floating pt hardware error';
410:C
411:C      4      otherwise
412:C      4      s := 'undocumented error'
413:C      4      end; (*CASE ERRORCODE*)
414:C      3      writeln(output,
415:C      3      '-----');
416:C      3      writeln('error ',errorcode:1,' ',s,cteol); write(cteol);
417:C      3      if excp_line >= 0 then write('line number: ',excp_line:1, ' ');
418:C      3      writeln('PC value: ',excp_pc:1);

```

```

419:C      3      end;
420:C      2      1: end (*PRINTERROR*);
421:C
422:C
423:C
424:D      -2      1      (*DIRECTORY ACCESS METHOD FOR DIRECT UNIT OPEN, ASSUMES NO DIRECTORY *)
425:D
426:D      1      procedure unblockeddam(anyvar f: fib; unum: unitnum; request: damrequesttype);
427:D      2      type vidptr = ^vid;
428:D      2      var buf: shortint;
429:D      2      begin
430:D      2      ioresult := 0;
431:D      2      with f, unitable[unum] do
432:D      3      case request of
433:C      4      stretchit,      {can't do anything}
434:C      4      purgefile, closefile:      {nothing to do};
435:C      4      {changed stripname for version 2.2 on 4-May-83}
436:C      4      stripname: begin      {move ftitle to ftid and set ftitle to null}
437:C      5      if strlen(ftitle) > tidlen then
438:C      5      begin
439:C      6      ioresult := ord(ibadtitle);
440:C      6      setstrlen(ftid,0);
441:C      5      end
442:C      5      else
443:C      5      ftid := ftitle;
444:C      5      setstrlen(ftitle,0);
445:C      4      end;
446:C      4      getvolumename: vidptr(addr(f))^ := uvid;
447:C      4      setvolumename: uvid := vidptr(addr(f));
448:C      4      openunit;
449:C      4      openvolume;
450:C      4      createfile;
451:C      4      openfile: begin
452:C      5      fileid := 0;
453:C      5      ureportchange := false;
454:C      5      if request = openunit then fpeof := umaxbytes
455:C      5      else if not uisblkd then fpeof := maxint
456:C      6      else begin
457:C      6      FPEOF := MAXINT;      {CHANGED FOR 9122 4/11/84 SFB}
458:C      6      call(tm, addr(f), readbytes, buf, 2, 0); {TOUCH DISC}
459:C      6      {UNITABLE[UNUM].UMAXBYTES WAS VALIDATED BY TM CALL -
460:C      6      IF DISC CONTROLLER SENT PRESENT MEDIA VALUE IF SMART
461:C      6      ENOUGH}
462:C      6      fpeof := ueovbytes(unum);      {MOVED 4/11/84 SFB}
463:C      6      {UEOVBYTES SET TO UMAXBYTES IF BLOCKED & NOT 988S/989S}
464:C      6      end;
465:C      4      ureportchange := true; umediavalid := true;
466:C      4      fleof := fpeof;
467:C      4      fisnew := false;
468:C      4      if not uisblkd then
469:C      5      if not fistextvar then am := tm
470:C      6      else am := serialtextamhook
471:C      6      else if not fbuffered then am := amtable[untypedfile]
472:C      6      else if not fistextvar then am := amtable[datafile]
473:C      7      else am := amtable[fkind];
474:C      4      end;
475:C
476:C      4      otherwise ioresult := ord(ibadrequest);
477:C      4      end;
478:C      2      end;

```



```

479:S
480:D -2 1 (* ACCESS METHOD FOR UNBUFFERED TRANSFERS *)
481:D -2 1 {updates fpos, checks logical limits of file, calls "stretch" if necessary}
482:S
483:D 1 procedure unbufferedam(fp: fibp; request: amrequesttype;
484:D 2 anyvar buffer: window; bufsize, position: integer);
485:D 2 label 1;
486:D 2 begin
487:C 2 with fp^, unitable^[funit] do
488:C 3 case request of
489:C 4 readbytes,writebytes:
490:C 4 begin
491:C 4 fpos := position + bufsize;
492:C 4 if fpos > fleof then
493:C 5 if (request=readbytes) then begin ioresult := ord(ieof); goto 1; end
494:C 5 else
495:C 6 begin
496:C 6 if fpos > fpeof then
497:C 7 begin
498:C 7 call(dam, fp^, funit, stretchit);
499:C 7 if fpos > fpeof then begin ioresult := ord(ieof); goto 1; end
500:C 8 end;
501:C 8 fleof := fpos; fmodified := true;
502:C 8 end;
503:C 4 call(tm, fp, request, buffer, bufsize, position);
504:C 4 end;
505:C 4 flush: call(tm, fp, request, buffer, bufsize, position);
506:C 4 otherwise ioresult := ord(lbadrequest);
507:C 4 end;
508:C 2 1;
509:C 2 end.
510:S
511:S
512:D -2 1 (* ACCESS METHOD FOR SERIAL TEXT INPUT DEVICES *)
513:D -2 1 {converts a carriage return character to an 'end of line' indication}
514:S
515:D 1 procedure serialtextam(fp: fibp; request: amrequesttype;
516:D 2 anyvar buffer: window; bufsize, position: integer);
517:D -4 2 var i: integer;
518:D 2 begin
519:C 2 with fp^, unitable^[funit] do
520:C 3 begin
521:C 3 call(tm, fp, request, buffer, bufsize, position);
522:C 3 if ioresult = ord(inoerror) then
523:C 4 if request = readbytes then
524:C 5 begin
525:C 5 feol := buffer[bufsize-1] = eol;
526:C 5 for i := 0 to bufsize - 1 do if buffer[i]=eol then buffer[i] := ' ';
527:C 5 end;
528:C 3 end;
529:C 2 end.
530:S
531:D -2 1 (* ACCESS METHOD FOR DATA FILES *)
532:D -2 1 {accomplishes general purpose buffering}
533:S
534:D 1 procedure standardam(fp: fibp; request: amrequesttype;
535:D 2 anyvar buffer: window; bufsize, position: integer);
536:D 2 label 1,2;
537:D -36 2 var lastblock, block, oldfleof, oldfpos, (rdq)
538:D firstpos, firstbytes, middlebytes, endbytes, i: integer;

```

```

539:D -37 2 c: char;
540:S
541:D 2 procedure flushbuffer;
542:D -4 3 var bufsize: integer;
543:C 3 begin with fp^ do
544:C 4 if fbufchanged {block buffer has been written} then
545:C 5 begin
546:C 5 bufsize := fleof - flastpos;
547:C 5 if bufsize > fblksize then bufsize := fblksize;
548:C 5 call(unitable^[funit].tm, fp, writebytes, fbuffer, bufsize, flastpos);
549:C 5 if ioresult <> ord(inoerror) then goto 1;
550:C 5 fbufchanged := false;
551:C 5 end;
552:C 3 end;
553:S
554:D 2 procedure fetchbuffer;
555:D -8 3 var i, bufsize: integer;
556:C 3 begin if block<>lastblock then with fp^ do
557:C 4 begin
558:C 4 flushbuffer;
559:C 4 lastblock := block; flastpos := lastblock*fblksize;
560:C 4 bufsize := oldfleof - flastpos;
561:C 4 if bufsize <= 0 then bufsize := 0 else
562:C 5 begin
563:C 5 if bufsize > fblksize then bufsize := fblksize;
564:C 5 call(unitable^[funit].tm, fp, readbytes, fbuffer, bufsize, flastpos);
565:C 5 if ioresult <> ord(inoerror) then goto 1;
566:C 5 end;
567:C 5 for i := bufsize to fblksize-1 do fbuffer[i] := chr(0);
568:C 5 end;
569:C 3 end;
570:S
571:C 2 begin with fp^ do
572:C 3 begin
573:C 3 ioresult := ord(inoerror);
574:C 3 oldfleof := fleof; oldfpos := fpos; (rdq)
575:C 3 case request of
576:C 4 flush: begin
577:C 4 flushbuffer;
578:C 4 call(unitable^[funit].tm, fp, flush, fp^, 0, 0);
579:C 4 end;
580:C 4 writeeol: begin
581:C 4 c := eol;
582:C 4 standardam(fp, writebytes, c, 1, position);
583:C 4 end;
584:C 4 readtoeol:
585:C 4 begin
586:C 4 middlebytes := 0;
587:C 4 fpos := position;
588:C 4 if bufsize > fleof - position then bufsize := fleof - position;
589:C 4 lastblock := (flastpos - flastpos mod fblksize) div fblksize;
590:C 4 while bufsize > 0 do
591:C 5 begin
592:C 5 block := fpos div fblksize;
593:C 5 fetchbuffer;
594:C 5 firstpos := fpos - flastpos;
595:C 5 firstbytes := fblksize - firstpos;
596:C 5 if firstbytes > bufsize then firstbytes := bufsize;
597:C 5 for i := 0 to firstbytes-1 do
598:C 6 if fbuffer[firstpos+i]=eol then

```



```

599:C      7      begin firstbytes := i; bufsize := i; goto 2; end;
600:C      5      2: moveleft(fbbuffer[firstpos], buffer[i+middlebytes], firstbytes);
601:C      5      middlebytes := middlebytes + firstbytes;
602:C      5      fpos := fpos + firstbytes;
603:C      5      bufsize := bufsize - firstbytes;
604:C      5      end;
605:C      4      buffer[0] := chr(middlebytes);
606:C      4      if middlebytes>0 then fp^.feoln := false;
607:C      4      end;
608:C      4      readbytes,writebytes:
609:C      4      begin
610:C      4      fpos := position + bufsize;
611:C      4      if fpos > ffeof then
612:C      5      if (request = readbytes) then
613:C      6      begin {feoln := true;} ioresult := ord(ieof); fpos:=oldfpos; goto 1; end
614:C      6      else begin
615:C      7      if fpos > fpeof then
616:C      8      begin
617:C      9      call(unitable^[funit].dam, fp^, funit, stretchit);
618:C      9      if fpos > fpeof then begin ioresult := ord(ieof); fpos := oldfpos;
619:C      9      goto 1;
620:C      9      end;
621:C      7      end;
622:C      6      ffeof := fpos; fmodified := true;
623:C      6      end;
624:C      4      if flastpos < 0 then lastblock := -1
625:C      5      else lastblock := flastpos div fblksize;
626:C      4      block := position div fblksize;
627:C      4      if (bufsize=1) and (block=lastblock) {the most common case!} then
628:C      5      if request = readbytes then buffer[0] := fbbuffer[position mod fblksize]
629:C      6      else begin
630:C      7      fbbuffer[position mod fblksize] := buffer[0];
631:C      8      fbchanged := true;
632:C      8      end
633:C      6      else
634:C      7      begin
635:C      8      firstpos := (-position) mod fblksize;
636:C      9      if firstpos >= bufsize then
637:C      10     begin firstbytes := bufsize; endbytes := 0; end
638:C      10     else begin firstbytes := firstpos; endbytes := fpos mod fblksize; end;
639:C      10     middlebytes := bufsize - firstbytes - endbytes;
640:C      10     end;
641:C      5      if firstbytes > 0 then
642:C      6      begin
643:C      7      fetchbuffer;
644:C      8      if request=readbytes
645:C      9      then moveleft(fbbuffer[fblksize-firstpos], buffer, firstbytes)
646:C      9      else
647:C      10     begin
648:C      11     moveleft(buffer, fbbuffer[fblksize-firstpos], firstbytes);
649:C      11     fbchanged := true;
650:C      11     end;
651:C      6      end;
652:C      6      block := block + 1;
653:C      6      end;
654:C      5      while middlebytes > 0 do
655:C      6      begin
656:C      7      if request=readbytes
657:C      8      then begin
658:C      9      fetchbuffer;

```

```

659:C      7      moveleft(fbbuffer, buffer[firstbytes], fblksize)
660:C      7      end
661:C      7      else begin
662:C      8      flushbuffer;
663:C      9      lastblock := block; flastpos := lastblock * fblksize;
664:C      9      moveleft(buffer[firstbytes], fbbuffer, fblksize);
665:C      9      fbchanged := true;
666:C      9      end;
667:C      6      middlebytes := middlebytes - fblksize;
668:C      6      firstbytes := firstbytes + fblksize;
669:C      6      block := block + 1;
670:C      6      end;
671:C      5      if endbytes > 0 then
672:C      6      begin
673:C      7      fetchbuffer;
674:C      8      if request=readbytes
675:C      9      then moveleft(fbbuffer, buffer[firstbytes], endbytes)
676:C      9      else begin
677:C      10     moveleft(buffer[firstbytes], fbbuffer, endbytes);
678:C      10     fbchanged := true;
679:C      10     end;
680:C      6      end;
681:C      5      end;
682:C      4      if flistextvar then if request = readbytes then
683:C      5      for i := 0 to bufsize-1 do
684:C      6      if buffer[i] = eol then begin feoln := true; buffer[i] := ' '; end
685:C      6      else feoln := false;
686:C      5      end;
687:C      4      otherwise ioresult := ord(ibadrequest);
688:C      4      end;
689:C      3      end;
690:C      2      1:
691:C      2      end;
692:C      2
693:C      2
694:C      2
695:C      2
696:C      2
697:C      2      serialtextamhook := serialtextam;
698:C      2
699:C      2      {new(efhtable); ALREADY 00NE IN BOOT LOADER}
700:C      2      new(amtable);
701:C      2      new(suffixtable);
702:C      2
703:C      2      for fk := untypedfile to lastfk do
704:C      3      begin
705:C      4      suffixtable^[fk] := ''; {no suffix}
706:C      4      amtable^[fk] := unbufferedam; {no buffering}
707:C      4      efhtable^[fk] := 0; {unassociated LIF file type}
708:C      4      end;
709:C      2
710:C      2      efhtable^ [untypedfile] := 3; {LIF directory}
711:C      2      {no suffix}
712:C      2      {no buffering}
713:C      2
714:C      2      suffixtable^[badfile] := 'BAD'; {bad block indication}
715:C      2      efhtable^[badfile] := 2; {LIF bad block marker}
716:C      2      {no buffering}
717:C      2
718:C      2      efhtable^ [datafile] := -5622; {OCD Pascal data file}

```



```

719:C      2      (no suffix)
720:C      2      amtable^ [datafile]      := standardam; (general purpose buffering)
721:S
722:C      2      suffixtable^[codefile]    := 'CODE';      {code file suffix}
723:C      2      efttable^[codefile]       := -5582;        {DCD Pascal code file}
724:C      2      (no buffering)
725:S
726:C      2      suffixtable^[sysfile]      := 'SYSTEM';    {suffix for system file}
727:C      2      efttable^[sysfile]         := -5822;      {DCD system (boot) file}
728:C      2      (no buffering)
729:C      2      end;
730:S
731:S
732:C      1      end {miscellaneous stuff module}
733:C
734:C      2      $include 'MISC'$;
2130:C      2      $LIST ON$
2131:S
2132:C      2      $LIST OFF, include 'FS'$;
2132:D      1      $LIST ON, include 'INITUNITS'$;
2133:D      1      module initunits;
2134:S
2135:D      1      import sysglobals,mini,asm,fs;
2136:S
2137:D      1      export
2138:S
2139:D      1      procedure nount(fp: fibp; request: amrequesttype; anyvar buffer: window;
2140:D      2      bufsize, position: integer);
2141:D      1      procedure nodam(anyvar f: fib; unum: unitnum; request: damrequesttype);
2142:D      1      procedure noisr(isribptr: pisrib);
2143:D      1      procedure unioinit;
2144:S
2145:D      1      implement
2146:S
2147:D      1      procedure crtio$alias 'sysdevs_crtio'$
2148:D      2      (fp: fibp; request: amrequesttype; anyvar buffer: window;
2149:D      2      bufsize, position: integer); EXTERNAL;
2150:S
2151:D      1      procedure kbdiio$alias 'sysdevs_kbdio'$
2152:D      2      (fp: fibp; request: amrequesttype; anyvar buffer: window;
2153:D      2      bufsize, position: integer); EXTERNAL;
2154:S
2155:D      1      procedure boottm $alias 'bootdammodule_boottm'$
2156:D      2      (fp: fibp; request: amrequesttype; anyvar buffer: window;
2157:D      2      bufsize, position: integer); EXTERNAL;
2158:S
2159:D      1      procedure bootdam $alias 'bootdammodule_bootdam'$
2160:D      2      (anyvar f: fib; unum: unitnum; request: damrequesttype); EXTERNAL;
2161:S
2162:D      1      procedure initbootdam $alias 'bootdammodule_initbootdam'$; EXTERNAL;
2163:S
2164:D      1      procedure unblockeddam $alias 'MISC_UNBLOCKEDDAM'$
2165:D      2      (anyvar f: fib; unum: unitnum; request: damrequesttype); EXTERNAL;
2166:S
2167:D      1      procedure noisr(isribptr: pisrib);
2168:C      2      begin
2169:C      2      end;
2170:S
2171:D      -4      1      procedure initsysisr;
2172:D      2      var i:integer;

```

```

2173:C      2      begin
2174:C      2      for i:= 1 to 7 do
2175:C      3      begin
2176:C      3      interrupttable[i] := nil;
2177:C      3      perminttable[i] := nil;
2178:C      3      end;
2179:C      2      end;
2180:S
2181:D      1      procedure initunitentry(un: unitnum;
2182:D      2      p_dam: damtype;
2183:D      2      p_am: amtype;
2184:D      2      {
2185:D      2      { p_sc, }
2186:D      2      { p_ba, }
2187:D      2      { p_du, }
2188:D      2      { p_dv: byte; }
2189:D      2      { p_byteoffset: integer; }
2190:D      2      { p_devid: shortint; }
2191:D      2      { p_uvid: vid; }
2192:D      2      { p_dvrtemp: integer }
2193:D      2      { p_letter: char; }
2194:D      2      { p_offline: boolean }
2195:D      2      { p_uisinteractive: boolean; }
2196:D      2      { p_umedialvalid: boolean; }
2197:D      2      { p_uppercase: boolean; }
2198:D      2      { p_uisfixed: boolean; }
2199:D      2      { p_uireportchange: boolean; }
2200:D      2      { p_pad: 0..1 }
2201:D      2      { p_uisblkd: boolean; }
2202:D      2      { p_umaxbytes: shortint } );
2203:C      2      begin (initunitentry)
2204:C      3      with unitable[un] do begin
2205:C      3      dam := p_dam;
2206:C      3      tm := p_am;
2207:C      3      sc := 0;
2208:C      3      ba := 0;
2209:C      3      du := 0;
2210:C      3      dv := 0;
2211:C      3      byteoffset := 0;
2212:C      3      devid := 0;
2213:C      3      uvid := p_uvid;
2214:C      3      dvrtemp := 0;
2215:C      3      letter := chr(0);
2216:C      3      offline := false;
2217:C      3      uisinteractive := p_uisinteractive;
2218:C      3      umediavalid := false;
2219:C      3      uppercase := not p_uisblkd;
2220:C      3      uisfixed := false;
2221:C      3      ureportchange := false;
2222:C      3      pad := 0;
2223:C      3      uisblkd := p_uisblkd;
2224:C      3      if uisblkd then
2225:C      4      umaxbytes := 0;
2226:C      3      end (with);
2227:C      3      end; (initunitentry)
2228:S
2229:D      1      procedure nount(fp: fibp; request: amrequesttype; anyvar buffer: window;
2230:D      2      bufsize, position: integer);
2231:S
2232:D      1      procedure nodam(anyvar f: fib; unum: unitnum; request: damrequesttype);

```



```

2233:C      2 begin ioresult := ord(znodevice) end;
2234:S
2235:D      1 procedure initunitable;
2236:D      2 var i: unitnum;
2237:C      2 begin
2238:C      2   new(unitable);
2239:C      2   for i := 0 to maxunit do
2240:C      3   initunitentry
2241:C      3     (i, nodam      , nounit, '',      'false,false);
2242:S
2243:C      2 initunitentry
2244:C      2   (1, unblockddam,crtio, 'CONSOLE', true, false);
2245:C      2 initunitentry
2246:C      2   (2, unblockddam,kbdio, 'SYSTEM',  false,false);
2247:C      2 initunitentry
2248:C      2   (3, bootdam      , boottm, '',      false,true );
2249:C      2 initunitentry
2250:C      2   (8, unblockddam,crtio, 'PRINTER',  false,false);
2251:S
2252:C      2 end;
2253:S
2254:D      1 procedure unitioinit;
2255:D      2 var i: unitnum;
2256:D      2   f: fib;
2257:C      2   begin
2258:C      2     initsysisr;
2259:S
2260:C      2   sysunit := 3;
2261:C      2   initunitable;
2262:C      2 end;
2263:S
2264:S      1 end
2265:C
2266:S
2267:C      2 $LIST ON, include 'INITUNITS';
2268:D      1 $include 'LDR';
2269:D      1 module ldr;
2270:S
2271:D      1 import sysglobals, misc, fs, loader, asm;
2272:D
2273:D      1 export
2274:D      1   procedure openlinkfile(var filename: string);
2275:S      1   procedure load(filetogo:fid; permanent: boolean);
2276:D
2277:D      1   procedure initsysunit;
2278:D      1   procedure lockfiles;
2279:S      1   procedure openfiles;
2280:D
2281:D      1   procedure go;
2282:S      1   procedure loadrom(name: fid);
2283:S
2284:D      1 implement
2285:S
2286:D      1 procedure lockfiles;
2287:C      2 begin
2288:C      2   close(input);
2289:C      2   close(output, 'LOCK');
2290:C      2   close(gfiles[2]);
2291:C      2   close(gfiles[4], 'LOCK');

```

```

{KEYBOARD}
{LISTING}

```

```

2282:C      2 end;
2283:S
2284:D      1 procedure openfiles;
2285:S      2 begin
2286:C      3   if not fibp(addr(input))^freadable then
2287:C      3     reset (input, 'CONSOLE:');
2288:C      3   with fibp(addr(output))^do
2289:C      3     begin
2290:C      3       if (fleaf>0) and fwriteable then close (output, 'LOCK');
2291:C      3       if not fwriteable then rewrite(output, 'CONSOLE:');
2292:C      3     end;
2293:S
2294:C      2   if not fibp(gfiles[2])^freadable then
2295:C      3     reset (gfiles[2]^, 'KEYBOARD');
2296:C      3   with fibp(gfiles[4])^do
2297:C      3     begin
2298:C      3       if (fleaf>0) and fwriteable then close (gfiles[4]^, 'LISTING');
2299:C      3       if not fwriteable then rewrite(gfiles[4]^, 'LOCK');
2300:C      3       if not fwriteable then rewrite(gfiles[4]^, 'PRINTER:LISTING.ASC');
2301:C      3     end;
2302:S
2303:D      2 end;
2304:S
2305:D      1 procedure initsysunit;
2306:D      2 var i: integer;
2307:C      2 begin
2308:C      2   syvid := '#'; strwrite(syvid,2,i,sysunit:1);
2309:C      2   i := findvolume(syvid, true);
2310:C      2   dkvid := syvid;
2311:C      2 end; (*INITUNITABLE*)
2312:S
2313:D      1 procedure go;
2314:D      2 var lastioresult:integer;
2315:D      2   userheap: anyptr;
2316:D      2   done: boolean;
2317:D      2   modptr: moddescptr;
2318:D      2   begin (G0)
2319:C      2   if entrypoint = nil then writeln('no start address')
2320:C      3   else
2321:C      3     begin
2322:C      3       page(output);
2323:C      3       mark(userheap);
2324:C      3       modptr := entrypoint;
2325:C      3       repeat
2326:C      4         done := modptr^.lastmodule;
2327:C      4         if modptr^.startaddr<>0 then
2328:C      5           begin
2329:C      5             userprogram(modptr^.startaddr,userstack); (** ALL PROGRAMS ARE ENTERED HERE **)
2330:C      5             if escapecode <> 0 then
2331:C      6               begin
2332:C      6                 lastioresult:=ioresult;(save ioresult)
2333:C      6                 release(userheap);
2334:C      6                 if escapecode <> -1 then printerror(escapecode,lastioresult);
2335:C      6                 done := true;
2336:C      6               end;
2337:C      5             end;
2338:C      4         modptr := modptr^.link;
2339:C      4         until done;
2340:C      3       end;
2341:C      2   end; (G0)
2342:S
2343:D      1 procedure loadrom(name: fid);
2344:S
2345:D      2 var lastioresult:integer;
2346:D      2   userheap: anyptr;
2347:D      2   done: boolean;
2348:D      2   modptr: moddescptr;
2349:D      2   begin (G0)
2350:C      2   if entrypoint = nil then writeln('no start address')
2351:C      3   else
2352:C      3     begin
2353:C      3       page(output);
2354:C      3       mark(userheap);
2355:C      3       modptr := entrypoint;
2356:C      3       repeat
2357:C      4         done := modptr^.lastmodule;
2358:C      4         if modptr^.startaddr<>0 then
2359:C      5           begin
2360:C      5             userprogram(modptr^.startaddr,userstack); (** ALL PROGRAMS ARE ENTERED HERE **)
2361:C      5             if escapecode <> 0 then
2362:C      6               begin
2363:C      6                 lastioresult:=ioresult;(save ioresult)
2364:C      6                 release(userheap);
2365:C      6                 if escapecode <> -1 then printerror(escapecode,lastioresult);
2366:C      6                 done := true;
2367:C      6               end;
2368:C      5             end;
2369:C      4         modptr := modptr^.link;
2370:C      4         until done;
2371:C      3       end;
2372:C      2   end; (G0)
2373:S
2374:D      1 procedure loadrom(name: fid);

```



```

2352:D -144 2 var modptr: moddescptr; upcname: tid;
2353:C 2 begin
2354:C 2   entrypoint := nil;
2355:C 2   if strlen(name) <= tidlength then
2356:C 3     begin
2357:C 3       upcname := name;   upc(upcname);
2358:C 3       modptr := sysdefs;
2359:C 3       while modptr <> nil do with modptr^ do
2360:C 5         begin
2361:C 5           if (startaddr < 0) and
2362:C 6             ((programe = name) or (ucase and (programe = upcname))) then
2363:C 6             begin
2364:C 6               entrypoint := modptr;
2365:C 6               modptr := nil;
2366:C 6             end
2367:C 6           else modptr := link;
2368:C 5         end;
2369:C 3       end;
2370:C 2   end;
2371:C 2
2372:D 1 procedure openlinkfile(var filename: string);
2373:C 2 var extra: addrec;
2374:C 2 begin (openlinkfile)
2375:C 2   getbytes(extra.a, sizeof(addrec)); {save old value of loadfib}
2376:C 2   extra.arp := loadfib;
2377:C 2   getbytes(loadfib.a, sizeof(fib,1)); {create a FIB for new file}
2378:C 2   finitb(loadfib.fbp, nil, -1); {untyped, unbuffered file}
2379:C 2   reset(loadfib.php, filename, 'shared');
2380:C 2   openlinkf(extra);
2381:C 2 end;
2382:S
2383:D 1 procedure printunres;
2384:D -4 2 var modptr: moddescptr;
2385:D -8 2 strptr: addrec;
2386:D -10 2 i: shortint;
2387:C 2 begin
2388:C 2   modptr := newmod;
2389:C 2   while modptr <> endmod do with modptr^ do
2390:C 4     begin
2391:C 4       if not resolved then
2392:C 4         for i := 0 to listsize - 1 do
2393:C 5           if listaddr[i] <> 0 then
2394:C 7             begin
2395:C 7               strptr.a := extaddr.a + listaddr[i];
2396:C 7               writeln(strptr.syp);
2397:C 7             end;
2398:C 4           modptr := link;
2399:C 4         end;
2400:C 2       writeln;
2401:C 2       writeln('The external references above were unresolved.');
```

```

2412:D -82 3 var s: string80;
2413:C 3 begin
2414:C 3   getioerrmsg(s, ioreult);
2415:C 3   writeln('cannot open ', fileto, '');
2416:C 3   writeln(s);
2417:C 3 end;
2418:C
2419:C 2 begin
2420:S
2421:S (*This routine must:
2422:S 1. Set "ENTRYPOINT" to nil (no start address) or correct value.
2423:S 2. Load executable code into heap.
2424:S 3. Set "SYSEFS" to user program symbol table.
2425:S 4. Set "USERSTACK" to initial SP value. *)
2426:S
2427:C 2 loadfib.php := nil;
2428:C 2 try
2429:C 3   releaseuser; {get rid of last program}
2430:C 3   mark(lowheap.p); highheap.a := userstack;
2431:C 3   startrelloc := lowheap.a;
2432:C 3   newmod := sysdefs; endmod := sysdefs;
2433:C 3   writeln; writeln('Loading ', fileto, '');
2434:C 3   openlinkfile(fileto);
2435:C 3   if fdirectory = nil then begin printwhy; escape(-1); end;
2436:C 3   for modnum := 1 to fdirectory^[0].dnumfiles do
2437:C 4     begin loadinfo(modnum, true, false); checkrev; end;
2438:C 3   allresolved := true; matchfile;
2439:C 3   highheap.a := userstack;
2440:C 3
2441:C 3   if not allresolved then
2442:C 4     begin
2443:C 4       fileto := syslibrary;
2444:C 4       if syvid <> '' then openlinkfile(fileto)
2445:C 4       else fdirectory := nil;
2446:C 4       if fdirectory = nil then printwhy
2447:C 4       else
2448:C 5         begin
2449:C 5           repeat
2450:C 6             libfound := false;
2451:C 6             modnum := fdirectory^[0].dnumfiles;
2452:C 6             while (modnum > 0) and not allresolved do
2453:C 6               begin
2454:C 7                 loadinfo(modnum, false, true);
2455:C 7                 checkrev;
2456:C 7                 modnum := modnum - 1;
2457:C 7               end;
2458:C 6             until allresolved or not libfound;
2459:C 5             highheap.a := userstack;
2460:C 5           end;
2461:C 4         end;
2462:C 3       end;
2463:S
2464:C 3   if not allresolved then printunres;
2465:S
2466:C 3   countcode;
2467:S
2468:C 3   highheap.a := userstack - totalglobal;
2469:C 3   if highheap.a < (aS - 32768) then escape(117);
2470:C 3   startglobal := userstack - aS;
2471:C 3   userstack := highheap.a;
```



```

2472:C      3  zeromem(highheap.p, totalglobal);
2473:C      3
2474:C      3  loadtext(true);
2475:S      3
2476:C      3  movedefs(startreloc+totalreloc);
2477:S      3
2478:C      3  lowheap.a := startdefs + totaldefs;
2479:C      3  release(lowheap.p);
2480:S      3
2481:C      3  if not permanent then
2482:C      4  if entrypoint = nil then
2483:C      5  begin writeln('no start address');
2484:C      5  escape(-1);
2485:C      5  end;
2486:S      3
2487:C      3  recover
2488:C      3  begin
2489:C      4  ecode := escapecode; icode := ioresult;
2490:C      4  closefiles;
2491:C      4  if ecode <> -1 then
2492:C      5  case ecode of
2493:C      6  100..105: begin
2494:C      7  writeln('can't link object at byte ', wrongbyte:1);
2495:C      7  writeln(' in text record ', wrongrec:1);
2496:C      7  writeln(' of module ', linkmodname.syp, '');
2497:C      7  end;
2498:C      7  110: writeln('symbol def's nested too deeply');
2499:C      7  111: writeln('improper link info format');
2500:C      7  112: writeln('not enough memory to load');
2501:C      7  116: writeln('', filefgo, ' not a codefile');
2502:C      7  117: writeln('available global space exceeded');
2503:C      7  118: writeln('incorrect version number');
2504:C      7  otherwise printerror(ecode, icode);
2505:C      7  end;
2506:C      4  releaseuser;
2507:C      4  escape(-1);
2508:C      4  end;
2509:C      2  end; (LOAD)
2510:S      1  end
2511:C      2  $include 'LDR'$;
2512:C      2
2513:S      1
2514:D      1  import sysglobals, asm, loader, misc, fs, initunits, ldr;
2515:S      1
2516:D      1  var keyboard: text;
2517:D      1  listing: text;
2518:S      1
2519:D      -4  1  i: integer;
2520:S      1
2521:D      1  procedure initfiles;
2522:C      2  begin
2523:C      2  (INPUT ) new(gfiles[0]);
2524:C      2  (OUTPUT ) new(gfiles[1]);
2525:C      2  (KEYBOARD) new(gfiles[2]);
2526:C      2  (LISTING) new(gfiles[4]);
2527:C      2  end; (*INITFILES*)
2528:D      -4  1
2529:C      1  procedure dummydebug(p1,p2,p3: integer); begin end;
2530:C      1  procedure dummycleario; begin end;
2531:S

```

```

2532:S
2533:S
2534:C      1  begin
2535:S      1
2536:C      1  initvects;          (SET UP VECTORS, INTERRUPT TABLE, ETC.)
2537:C      1
2538:S      1  (*****
2539:C      1  MISCELLANEOUS LOW LEVEL STUFF)
2540:S      1
2541:C      1  idle:=250;          ( 3.0 bug fix -- assume old char rom 3/20/84)
2542:C      1  locklevel := 0; actionspending := 0;
2543:C      1  debugger := dummydebug;
2544:C      1  cleariohook := dummycleario;
2545:C      1  gfiles[1] := nil;
2546:S      1
2547:S      1  (*****
2548:C      1  INITIALIZE SUFFIX AND ACCESS METHOD TABLES)
2549:S      1
2550:C      1  initfilekinds;
2551:S      1
2552:S      1  (*****
2553:C      1  initialize ISR's,
2554:C      1  DEVICE DRIVERS,
2555:C      1  ACCESS METHODS,
2556:C      1  DIRECTORY ACCESS,
2557:C      1  DEFAULT UNIT TABLE,
2558:C      1  CLEAR UNITS )
2559:S      1
2560:C      1  unitioinit;
2561:S      1
2562:S      1  (*****
2563:C      1  FIND SYSTEM AND DEFAULT VOLUME NAMES)
2564:S      1
2565:C      1  initsysunit;
2566:S      1
2567:S      1  (*****
2568:C      1  create files INPUT, OUTPUT, KEYBOARD and LISTING)
2569:S      1
2570:C      1  initfiles;
2571:C      1  openfiles;
2572:S      1
2573:S      1  (*****
2574:C      1  INITIALIZE LOADER)
2575:C      1
2576:C      1  markuser;
2577:C      1  syslibrary := 'LIBRARY';
2578:C      1
2579:C      1  end. (module initsys)
2580:S      1
2581:C      1

```

No errors. 13 warnings.

***** Nonstandard language features enabled *****

INITLOAD

Description

INITLOAD is system-wide type and variable declarations, interface to internal minifloppy driver, boot-load routines, the linking loader and the boot-up control program.

Usage

INITLOAD is part of SYSTEM_P.

Notes

Includes: GLOBALS, MINI, BOOTDAM, LOADER and INITLOAD.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:D 0 $modcal, range off, ovfcheck off, debug off, iocheck off, stackcheck off$
2:D 0 program initload;
3:S
4:D I $search 'ASM' $
5:D
6:D I $include 'GLOBALS'S;
7:S I module sysglobals;
8:D
9:D 1 export
10:D
11:D I type fsidctype = packed array[1..20] of char;
12:D I const
13:D 1 $sidc = fsidctype['Rev. 3.01 20-Sep-84'];
14:D 1 $minint = -32768; (*20 CHARS: VERSION,DATE,TIME OF FILE SYS*)
15:D 1 $maxint = 32767; (*MINIMUM SHORT INTEGER VALUE*)
16:D 1 $maxunit = 50; (*MAXIMUM SHORT INTEGER VALUE*)
17:D 1 $passleng = 16; (*MAXIMUM PHYSICAL UNIT NUMBER*)
18:D 1 $vidleng = 16; (*NUMBER OF CHARS IN A PASSWORD*)
19:D 1 $vidleng = 16; (*NUMBER OF CHARS IN A VOLUME NAME*)
20:D 1 $fidleng = 12; (*NUMBER OF CHARS IN A FILE TITLE*)
21:D 1 $fbksize = 512; (*NUMBER OF CHARS IN FILE NAME*)
22:D 1 $maxsc=63; (*STANDARD FILE BUFFER LENGTH*)
23:D 1 $minlevel=1; (*LARGEST SELECT CODE *)
24:D 1 $maxlevel=6; (*LOWEST INTERRUPT LEVEL*)
25:S
26:S
27:S
28:S
29:S 1 type
30:D 1 byte = 0..255;
31:D 1 shortint = $minint..$maxint;
32:D 1 ipointer = ^integer;
33:D 1 charptr = ^char;
34:D 1 textptr = ^text;
35:D 1 string80 = string[80];
36:D 1 string255 = string[255];
37:D 1 suffixtype = string[5];
38:S 1 string255ptr = ^string255;
39:S
40:D 1 unitnum = 0..$maxunit;
41:D 1 vid = string[vidleng];
42:D 1 id = string[vidleng];
43:D 1 fid = string[fidleng];
44:D 1 passtype = string[passleng];
45:S
46:D 1 filekind = (untypedfile, (directory entry)
47:D 1 badfile, (bad blocks)
48:D 1 codefile, (executable or linkable)
49:D 1 textfile, (UCSD format text with editor environment)
50:D 1 ascfifile, (L.I.F. ASCII format text strings)
51:D 1 datafile, (file of <data type, e.g. char, integer,etc.>)
52:D 1 sysfile, (system (BOOT) file)
53:D 1 fkind7, fkind8, fkind9,
54:D 1 fkind10, fkind11, fkind12,
55:D 1 fkind13, fkind14, lastfkind); (*reserved for future expansion*)
56:S
57:S
58:D 1
59:S (*FILE INFORMATION*)

```

```

60:D 1 window = packed array [0..$maxint] of char;
61:D 1 windowp = ^window;
62:S
63:S
64:D 1 fitp = ^fib;
65:S
66:D 1 amrequesttype = (readbytes, writebytes, flush, writeeol, readtoeol, clearunit,
67:D 1 setcursor, getcursor, startread, startwrite, unitstatus, seekeof);
68:D 1 amtype = procedure (fp: fibp; request: amrequesttype;
69:D 1 anyvar buffer: window; bufsize, position: integer);
70:D 1 eotproc = procedure (fp: fibp);
71:S
72:D 1 fit = packed record
73:D 1 fwindow: windowp; (*BUFFER VARIABLE...F^*)
74:D 1 flistptr: fibp; (* (* LIST OF OPEN FILES *)*)
75:D 1 (*declaration and type information*)
76:D 1 frecsiz: integer; (* (* SIZE OF ONE LOGICAL RECORD *)*)
77:D 1 feft: shortint; (* (* EXTERNAL FILE TYPE *)*)
78:D 1 fkind: filekind; (* (* FILE KIND *)*)
79:D 1 fistextvar: boolean; (* (* FILE IS LINE FORMATTED *)*)
80:D 1 fbuffered: boolean; (* (* HAS 512 BYTE BLOCK BUFFER *)*)
81:D 1 fanonymous: boolean; (* (* FILE HAS NO NAME *)*)
82:D 1 fisnew: boolean; (* (* WAS CREATED THIS ASSOCIATION *)*)
83:D 1 freadable, fwritable: boolean; (* (* FILE ACCESS RIGHTS *)*)
84:D 1 (*state information*)
85:D 1 freadmode, fbuffvalid: boolean; (* (* F^ AND LOOKAHEAD STATES *)*)
86:D 1 feoln: boolean; (* (* F^ IS AN END OF LINE *)*)
87:D 1 feof: boolean; (* (* TRIED TO READ PAST END OF FILE *)*)
88:D 1 fmodified: boolean; (* (* FILE HAS CHANGED SIZE *)*)
89:D 1 fbuchanged: boolean; (* (* BUFFER NEEDS TO BE WRITTEN *)*)
90:D 1 (*file size and position*)
91:D 1 fpos: integer; (* (* FILE POINTER, CURRENT FILE POSITION *)*)
92:D 1 fleof: integer; (* (* LOGICAL END OF FILE, CURRENT FILE SIZE *)*)
93:D 1 fpeof: integer; (* (* PHYSICAL END OF FILE, MAXIMUM FILE SIZE *)*)
94:D 1 (*buffering and low level formatting information*)
95:D 1 flastpos: integer; (* (* FILE POSITION OF BUFFER *)*)
96:D 1 freptcnt: shortint; (* (* SPACE COMPRESSION COUNT *)*)
97:D 1 am: amtype; (* (* BUFFER METHOD MODULE *)*)
98:D 1 (*file association info*)
99:D 1 fstartaddress: integer; (* (* EXECUTION ADDRESS IN BOOT FILE *)*)
100:D 1 fvid: vid; (* (* VOLUME NAME *)*)
101:D 1 ffpw: passstype; (* (* FILE PASSWORD *)*)
102:D 1 fid: fid; (* (* FILE NAME *)*)
103:D 1 fpathid: integer; (* (* ADDITIONAL SYSTEM DEPENDENT INFORMATION *)*)
104:D 1 fanoncir: shortint; (* (* TEMP FILE IDENTIFIER *)*)
105:D 1 foptstring: string255ptr; (* (* OPTIONAL STRING PARAM *)*)
106:D 1 (*byte block transfer information*)
107:D 1 fileid: integer; (* (* START BYTE OF FILE, OR OTHER IDENTIFICATION *)*)
108:D 1 fb0,fb1, (* (* FOR FUTURE EXPANSION *)*)
109:D 1 fnostrtemp, (* (* TRUE IF NO SRM TEMP FILE CREATED *)*)
110:D 1 fwaitonlock, (* (* TRUE IF SRM SHOULD WAIT FOR LOCK *)*)
111:D 1 fpurgeoldlink, (* (* TRUE IF OLD SRM LINK IS TO BE PURGED *)*)
112:D 1 foverwritten, (* (* TRUE IF OPENED WITH OVERWRITE *)*)
113:D 1 fsavepathid, (* (* TRUE IF PATHID NOT UNIQUE TO FILEID *)*)
114:D 1 flockable, (* (* TRUE IF FILE OPENED AS LOCKABLE *)*)
115:D 1 flocked, (* (* TRUE IF FILE IS LOCKED *)*)
116:D 1 fbusy: boolean; (* (* TRUE IF DRIVER IS ACTIVE *)*)
117:D 1 funit: unitnum; (* (* PHYSICAL UNIT NUMBER *)*)
118:D 1 feet: eotproc; (* (* CALLED WHEN TRANSFER COMPLETES *)*)
119:D 1 fxpos: integer; (* (* X POSITION FOR GOTOTOX *)*)

```



```

120:D      1      ftypos      : integer;      (* Y POSITION FOR GOTDXY *)
121:D      1      foldfileid : integer;      (* FILEID FOR OLD SRM FILE ON REWRITE *)
122:D      1      {for future expansion}
123:D      1      fextra: array[0..2] of integer;
124:D      1      fextra2: shortint;
125:D      1      {large miscellaneous fields sometimes present}
126:D      1      case integer of
127:D      1          0: { (minimal F18 ends here) };
128:D      1          1: { ftitle: fid; } (* FILE NAME, EXCEPT VOLUME AND SIZE *)
129:D      1          2: { fbuffer: packed array [0..fbksize-1] of char; }
130:D      1      end (*F18*);
131:S
132:S
133:D      1      damrequesttype = (getvolumename, setvolumename, getvolumedate, setvolumedate,
134:D      1      changename, purgename,
135:D      1      openfile, createfile, overwritefile, closefile, purgefile,
136:D      1      stretchit, makedirectory, crunch, opendirirectory, closedirectory, catalog,
137:D      1      stripname, setunitprefix, openvolume, duplicatelink, openparentdir,
138:D      1      catpasswords, setpasswords, lockfile, unlockfile, openunit);
139:S
140:S
141:D      1      damtype = procedure (anyvar f: fib; unum: unitnum; request: damrequesttype);
142:S
143:D      1      unitentry = {unitable entry definition}
144:D      1      packed record
145:D      1          dam: damtype;      {directory access method}
146:D      1          tm: amtype;      {byte block transfer method}
147:D      1          sc: byte;      {select code}
148:D      1          ba: byte;      {bus address}
149:D      1          du: byte;      {disc unit}
150:D      1          dv: byte;      {disc volume}
151:D      1          byteoffset: integer; {physical starting byte of volume}
152:D      1          devid: integer;      {identifier (Amigo identify sequence)}
153:D      1          uvid: vid;      {volume id}
154:D      1          dvrtmp: integer;      {temp for driver use only; init to 0!}
155:D      1          dvrtmp2: shortint;      {temp for driver use only; init to 0!}
156:D      1          letter: char;      {device specifier letter}
157:D      1          offline: boolean;      {unit absent or down flag}
158:D      1          uisinteractive: boolean; {user can edit input}
159:D      1          umediavalid: boolean; {medium not changed since last access}
160:D      1          uppercase: boolean; {volume name must be uppercased}
161:D      1          uisfixed: boolean; {fixed/removeable media flag}
162:D      1          ureportchange: boolean; {driver mode: report/ignore media change}
163:D      1          pad: 0..1; { (bit not used yet) }
164:D      1          case uisbikd: boolean of
165:D      1              true: {umaxbytes: integer} {volume size in bytes}
166:D      1          end; {unitentry}
167:S
168:D      1      unitabletype = array [unitnum] of unitentry;      (*0 NOT USED*)
169:D      1      amtabletype = array [filekind] of amtype;
170:D      1      suftabletype = array [filekind] of suffixtype;
171:D      1      efttabletype = array [filekind] of shortint;
172:S
173:D      1      unitableptr = ^unitabletype;
174:D      1      amtableptrtype = ^amtabletype;
175:D      1      suftableptrtype = ^suftabletype;
176:D      1      efttableptrtype = ^efttabletype;
177:S
178:D      1      iorsltwd = { *note* the ioresult enumerations have been partitioned into two
179:S

```

```

180:S      1      mutually-exclusive groups: those beginning with 'z' are reserved
181:S      1      for the low-level drivers , and those beginning
182:D      1      with 'i' are reserved for the higher-level routines.)
183:S
184:D      1      {inoerror, zbadblock, ibadunit, zbadmode, ztimeout,
185:D      1      ilostunit, ilostfile, ibadtitle, inoroom, incount,
186:D      1      inofile, idupfile, inotclosed, inotopen, ibadformat,
187:D      1      znosuchblk, znodevice, zinitfail, zprotected,
188:D      1      zstrange1, zbadhardware, zcatchall, zbadhma,
189:D      1      inotvalidsize, inotreadable, inotwriteable, inotdirect,
190:D      1      idirfull, istrovfl, ibadclose, eof,
191:D      1      zunitialized, znoblock, znoready, znomedium,
192:D      1      inodirectory, ibadfiletype, ibadvalue, icantstretch,
193:D      1      ibadrequest, inotlockable, ifilelocked, ifileunlocked,
194:D      1      idirnotempty, itoomanyopen, ibadaccess, ibadpass, ifilenotdir,
195:D      1      inotondir, ineedtempdir, isrmcatchall, zmediumchanged,
196:D      1      {end marker} endioerrs);
197:S
198:D      1      proctype=procedure;
199:D      1      scstype=0..maxsc;
200:D      1      leveltype=minlevel..maxlevel;
201:D      1      pisrib = ^isrib;
202:D      1      isrproctype = procedure(isribptr: pisrib);
203:D      1      isrib = {isr information block}
204:D      1      packed record
205:D      1          intregaddr: charptr;      {interrupt register address}
206:D      1          intregmask: byte;      {interrupt register mask}
207:D      1          intregvalue: byte;      {interrupt register target value after masking}
208:D      1          chainflag: boolean;      {chaining flag}
209:D      1          proci: isrproctype;      {isr}
210:D      1          link: pisrib;      {pointer to next isrib in linked list}
211:D      1      end;
212:D      1
213:D      1      inttabletype = array [1..7] of pisrib;
214:S
215:D      1      action = procedure;
216:S
217:D      1      daterec      = packed record
218:D      1          year      : 0..100;      (*100 IS TEMP DISK FLAG*)
219:D      1          day      : 0..31;      (*DAY OF MONTH*)
220:D      1          month     : 0..12;      (*0 ==> DATE NOT MEANINGFUL*)
221:D      1      end;
222:S
223:D      1      timerec      = packed record
224:D      1          hour      : 0..23;
225:D      1          minute    : 0..59;
226:D      1          centisecond: 0..5999;
227:D      1      end;
228:S
229:D      1      datetimerec   = packed record
230:D      1          date      : daterec;
231:D      1          time      : timerec;
232:D      1      end;
233:S
234:D      1      var
235:D      1      {***** MANY OF THE FOLLOWING HAVE HARDCODED
236:D      1      ADDRESSES IN COMPILER AND ELSEWHERE *****}
237:S
238:D      1      {*** ERROR RECOVERY ***}
239:D      -2      sysescapecode: shortint;

```



```

240:D -6 1 openfileptr: anyptr;
241:D -10 1 recoverblock: anyptr;
242:S
243:D -10 1 (** MEMORY MANAGEMENT **)
244:D -14 1 heapmax: anyptr;
245:D -18 1 heapbase: anyptr;
246:S
247:D -18 1 (** I/O DRIVERS **)
248:D -22 1 sysioreresult: integer;
249:D -26 1 hardwarestatus: integer;
250:D -30 1 locklevel: integer;
251:D -34 1 unitable: unitableptr;
252:D -62 1 interrupttable: inttabletype;
253:D -66 1 endisrhook: integer;
254:D -70 1 actionspending: integer;
255:S
256:D -70 1 (** FILE SYSTEM **)
257:D -94 1 gfiles: array[0..5] of textptr;
258:S ( [0] INPUT [3] (unused)
259:S [1] OUTPUT [4] LISTING
260:D -94 1 [2] KEYBOARD [5] (unused) )
261:S
262:D -98 1 amtable: amtableptrtype; (pointer to access methods)
263:D -102 1 suffixtable: suftableptrtype; (pointer to list of suffixes)
264:D -106 1 efttable: efttableptrtype; (pointer to LIF file types)
265:D -110 1 sysunit: integer;
266:D -146 1 syvid,dkvid: vid; (*SYSUNIT VOLID & DEFAULT VOLID*)
267:D -268 1 syslibrary: fid;
268:S
269:D -268 1 (** DEBUGGER HOOK **)
270:D -276 1 debugger: procedure(p1, p2, p3: integer);
271:S
272:D -276 1 (** CLEAR I/O HOOK **)
273:D -284 1 cleariohook: procedure;
274:S
275:D -312 1 perminttable: inttabletype;
276:S
277:D -392 1 deferredaction: array[1..10] of action;
278:S
279:D -400 1 serialtextamhook: amtype; (access method for serial devices)
280:S
281:D -400 1 sysname[-574]: packed array[1..10] of char;
282:S
283:D -400 1 sysflag[-302]: packed record
284:D -400 1 reserved1, reserved2, nointhpib, crtconfigreg,
285:D -400 1 nokeyboard, highlightskorbiggraphics, biggraphics,
286:D -400 1 alpha50: boolean;
287:D -400 1 end;
288:S
289:D -400 1 sysflag2[hex('FFFEA')]: packed record
290:D -400 1 pad7to1 : 0..127;
291:D -400 1 prompresent: boolean;
292:D -400 1 end;
293:S
294:S endsysvars : shortint; (must be last thing
295:D -402 1 in SYSGLOBALS)
296:S
297:D -402 1 implement
298:C 1 end
299:C 2 $include 'GLOBALS'$;

```

```

299:D 1 $include 'MINI' $;
300:D 1 module mini;
301:S
302:S
303:D 1 import
304:D 1 sysglobals, asm (, misc);
305:S
306:S
307:D 1 export
308:S
309:D 1 procedure minio(fp: fibp; request: amrequesttype; anyvar buffer: window;
310:D 2 length, position: integer);
311:D 1 procedure ioresc(ioresult_value: iorsltwd);
312:S
313:D 1 function iorval: iorsltwd;
314:S
315:D 1 implement
316:S
317:S
318:D 1 var
319:D -2 1 most_recent_escapecode: shortint; (for post-mortem diagnostic purposes only!!!)
320:S
321:S
322:D 1 procedure ioresc(ioresult_value: iorsltwd);
323:C 2 begin (ioresc)
324:C 2 ioresult := ord(ioresult_value);
325:C 2 escape(-10)
326:C 2 end; (ioresc)

```



```

327:D -2 1 $page$
328:S
329:D 1 function iorval: iorsltwd;
330:S
331:D 2
332:D 2   type
333:D 2     te = (table entry type)
334:D 2     record
335:D 2       b(basic error): shortint;
336:D 2       p(pascal error): iorsltwd
337:D 2     end;
338:D 2     ect_type = (error conversion table type)
339:D 2     array[1..29] of te;
340:D 2
341:D 2   const
342:D 2     error_conversion_table = ect_type[
343:D 2       te[b: 1066, p: zinitfail], (bad track 0, side 0)
344:D 2       te[b: 2066, p: zinitfail], (more than 4 spares)
345:D 2       te[b: 3066, p: zbadhardware], (write fault or lost data)
346:D 2       te[b: 4066, p: znomedium], (timeout during init)
347:D 2       te[b: 1080, p: znomedium], (no media or door open)
348:D 2       te[b: 2080, p: znomedium], (no media or door open)
349:D 2       te[b: 3080, p: znomedium], (no media or door open)
350:D 2       te[b: 8080, p: zmmediumchanged], (media changed)
351:D 2       te[b: 9080, p: znomedium], (media changed during operation)
352:D 2       te[b: 1081, p: znoblock], (track not found)
353:D 2       te[b: 2081, p: zbadhardware], (restore error)
354:D 2       te[b: 3081, p: zbadhardware], (track 0 not found after reset)
355:D 2       te[b: 4081, p: zbadhardware], (read lost data error)
356:D 2       te[b: 5081, p: zbadhardware], (write lost data error or fault)
357:D 2       te[b: 6081, p: zbadhardware], (address lost data error)
358:D 2       te[b: 7081, p: znoblock], (address CRC error during write)
359:D 2       te[b: 1083, p: zprotected], (write protect error)
360:D 2       te[b: 2083, p: zprotected], (write protect error)
361:D 2       te[b: 1084, p: znoblock], (read record not found/d bit set)
362:D 2       te[b: 2084, p: znoblock], (write record not found)
363:D 2       te[b: 3084, p: znoblock], (address (track) not found)
364:D 2       te[b: 1087, p: znoblock], (address CRC error)
365:D 2       te[b: 1088, p: zbadblock], (read CRC error)
366:D 2       te[b: 6090, p: zstrangei], (unexpected interrupt)
367:D 2       te[b: 7090, p: zstrangei], (INT during wrt trk handshaking)
368:D 2       te[b: 8090, p: znomedium], (timeout waiting for interrupt)
369:D 2       te[b: 9090, p: zcatchall], (INT mask > 2 (flpy locked out))
370:D 2       te[b: 11090, p: zbadhardware], (timeout; floppy not responding)
371:D 2       te[b: 1082, p: znodevice], (2nd drive not present)
372:D 2
373:D -2 2   var
374:S     i: shortint;
375:C
376:C   begin (iorval)
377:C     most_recent_escapecode := escapecode; (for post-mortem diagnostic purposes only!!)
378:C     iorval := inoerror; (value returned if unrecognized)
379:C     for i := 1 to 29 do
380:C       with error_conversion_table[i] do
381:C         if escapecode=b then iorval := p;
382:C     end; (iorval)

```

```

382:D -2 1 $page$
383:S
384:D 1 procedure minio(fp: fibp; request: amrequesttype; anyvar buffer: window;
385:D 2   length, position: integer);
386:D 2
387:D 2   const
388:D 2     bootrev1 = chr(255);
389:D 2
390:D 2   var
391:D 2     bootrevflag[318]: char;
392:D -4 2     minidrive[-301]: packed record num: 0..255 end;
393:D -8 2     buf: charptr;
394:S     uep: ^unitentry;
395:D 2
396:D 2   procedure clear_unit;
397:C   begin (clear_unit)
398:C     try
399:C       f_pwr_on;
400:C       recover
401:C       if iorval<>inoerror
402:C         then ioresc(iorval)
403:C         else escape(escapecode);
404:C     end; (clear_unit)
405:S
406:D 2   procedure transfer(sector: integer);
407:S
408:D -1 3   var
409:D -6 3     transfer_successful: boolean;
410:D -10 3     whole_sectors: integer;
411:D -10 3     whole_sector_bytes: integer;
412:D -12 3     fubuffer[-658]: packed array[0..255] of byte;
413:D -14 3     saved_escapecode: shortint;
414:D -14 3     ior: iorsltwd;
415:S
416:C 3   begin (transfer)
417:S 3     repeat
418:C 4       try
419:C 4         whole_sectors := length div 256;
420:C 5         if whole_sectors>0 then
421:C 6           begin
422:C 7             case request of
423:C 7               readbytes, startread:
424:C 7                 flpymread(whole_sectors, sector, buf^);
425:C 7               writebytes, startwrite:
426:C 7                 flpymwrite(whole_sectors, sector, buf^);
427:C 7             end; (case)
428:C 6             sector := sector+whole_sectors;
429:C 6             whole_sector_bytes := whole_sectors*256;
430:C 6             buf := addr(buf, whole_sector_bytes);
431:C 6             length := length-whole_sector_bytes;
432:C 6             end; (if)
433:C 5         if length>0 then
434:C 6           case request of
435:C 6             readbytes, startread:
436:C 7               begin
437:C 7                 flpyread(sector, fubuffer);
438:C 7                 moveleft(fubuffer, buf^, length);
439:C 7               end;
440:C 6             writebytes, startwrite:
441:C 7               begin

```



```

442:C      7      end;
443:C      7      writebytes, startwrite:
444:C      7      begin
445:C      7          moveleft(buf^, fubuffer, length);
446:C      7          flpy_wrt(sectors, fubuffer);
447:C      7      end;
448:C      7      end; {case}
449:S      7
450:C      5      transfer_successful := true;
451:S      5
452:C      5      recover
453:C      5      begin
454:C      5          ior := iorval;
455:C      5          if ior=inoerror then {unrecognized escapecode}
456:C      5              begin
457:C      5                  saved_escapecode := escapecode;
458:C      5                  try
459:C      5                      pwr_on;
460:C      5                      recover
461:C      5                      {do nothing};
462:C      5                  escape(saved_escapecode);
463:C      5              end
464:C      5              else if ior=zmediumchanged then
465:C      5                  begin
466:C      5                      uep^.umediavalid := false;
467:C      5                      if uep^.ureportchange then
468:C      5                          ioresc(ior);
469:C      5                      transfer_successful := false; {retry}
470:C      5                  end {with}
471:C      5              else
472:C      5                  ioresc(ior);
473:C      5              end; {recover}
474:S      5
475:C      4      until transfer_successful;
476:S      4
477:C      3      end; {transfer}

```

```

478:D      -8 2  $page$
479:S      2
480:C      2      begin {minilo}
481:C      2          uep := addr(unitable^[fp^.funit]);
482:C      2          if uep^.offline then
483:C      3              ioresult := ord(znodevice)
484:C      3          else
485:C      3              begin
486:C      3                  {lockup;}
487:C      3                  try
488:C      4                      if bootrevflag=bootrev1 then
489:C      5                          if uep^.du=0 then {ok}
490:C      6                              else ioresc(znodevice)
491:C      6                          else
492:C      5                              if uep^.du in [0,1] then minidrive.num := uep^.du
493:C      6                              else ioresc(znodevice);
494:S      6
495:C      4          case request of
496:C      5              clearunit:
497:C      5                  clear_unit;
498:S      5
499:C      5              unitstatus:
500:C      5                  fp^.fbusy := false;
501:S      5
502:C      5              flush:
503:C      5                  {do nothing};
504:S      5
505:C      5              readbytes, writebytes, startread, startwrite:
506:C      5              begin
507:C      5                  if uep^.ureportchange and not uep^.umediavalid then
508:C      6                      ioresc(zmediumchanged);
509:C      6                      buf := addr(buffer);
510:C      6                      if (position mod 256<>0) or odd(integer(buf)) then
511:C      6                          ioresc(zbadmode);
512:C      6                      if (position<0) or (length<0) or (position+length>fp^.fpeof) then
513:C      6                          ioresc(ieof);
514:C      6                      transfer[(position+fp^.fileid+uep^.byteoffset) div 256];
515:C      5                  end;
516:S      5
517:C      5              otherwise
518:C      5                  ioresc(ibadrequest);
519:C      5              end; {case}
520:S      5
521:C      4          ioresc(inoerror); {set ioresult & perform lockdown}
522:C      4          recover
523:C      4          begin
524:C      4              {lockdown;}
525:C      4              if escapecode<>-10 then escape(escapecode);
526:C      4              if request in [startread, startwrite] then call (fp^.feot, fp);
527:C      4              end; {recover}
528:C      3          end; {else}
529:C      2      end; {minilo}
530:S      2
531:C      1      end {mini}
532:S      1
533:C      2      $include 'MINI' $;
533:D      1      $include 'BOOTDAM'$;
534:D      1      module bootdammodule;
535:D      1      import sysglobals,asm,mini;
536:D      1      export

```



```

537:S
538:D      1  type string2=string[2];
539:S
540:D      -52 1  var nodestr: string[50];
541:S
542:D      -52 1  procedure initbootdam;
543:D      -52 1  procedure bootdam(anyvar f:fib; unum:unitnum; request:damrequesttype);
544:D      -52 1  procedure boottm(f:fibp; request:amrequesttype;
545:D      -52 2  anyvar buffer: window; bufsize,position: integer);
546:D      -52 1  function srmnode(sc: shortint): string2;
547:D      -52 1  function bootnode: string2;
548:S
549:D      -52 1 implement
550:D      -52 1 type
551:S
552:D      -52 1  strptr = ^string255;
553:D      -52 1  lifname = packed array[1..10] of char;
554:D      -52 1  word15 = 0..32767;
555:D      -52 1  bcd = 0..15;
556:D      -52 1  tdate = packed array[1..12] of bcd;
557:D      -52 1  direntry= ^direntry;
558:D      -52 1  direntry = packed record
559:D      -52 1      fname : lifname;
560:D      -52 1      ftype : shortint;
561:D      -52 1      fstart : integer;
562:D      -52 1      fsize : integer;
563:D      -52 1      fdate : tdate;
564:D      -52 1      lastvol : boolean;
565:D      -52 1      volnumber: word15;
566:D      -52 1      extension: integer;
567:D      -52 1  end;
568:S
569:D      -52 1  msustype = packed record
570:D      -52 1      mtype : byte;
571:D      -52 1      munit : byte;
572:D      -52 1      mscode: byte;
573:D      -52 1      maddr : byte;
574:D      -52 1  end;
575:D      -52 1  var
576:D      -52 1  driverkey [-293 {$FFFFE0B}]: boolean;
577:D      -52 1  boot_msus [-292 {$FFFFE0C}]: msustype;
578:D      -52 1  boot_id [16382 {$ 3FFE}]: shortint;
579:D      -52 1  boot_flags [16380 {$ 3FFC}]: packed array[0..7] of boolean;
580:S
581:D      -53 1  read : boolean;
582:S
583:D      -53 1  ( FUNCTIONS AND PROCEDURES USED FROM THE BOOT ROM )
584:D      -53 1  procedure boot_lifhead(var dsize,dstart:integer); external;
585:D      -53 1  procedure boot_findfile(var entry:direntyp; var lname:lifname;
586:D      -53 2  di,ds : integer); external;
587:D      -53 1  function boot_minit(msus:msustype):boolean; external;
588:D      -53 1  function boot_mfopen(anyvar fname:string255; var xaddr,length:integer;
589:D      -53 2  var ftype:shortint):boolean; external;
590:D      -53 1  procedure boot_mfclose; external;
591:D      -53 1  procedure boot_mread(secton,bytecount: integer;
592:D      -53 2  anyvar buffer>window; media:boolean); external;
593:S
594:D      1  function escio(escn:integer):iorsltd;
595:D      2  begin
596:D      2  case escn of

```

```

597:C      3  1,7 : escio:=znoddevice;
598:C      3  2 : escio:=znomedium;
599:C      3  3 : escio:=znotready;
600:C      3  4 : escio:=zbadblock;
601:C      3  5 : escio:=zbadhardware;
602:C      3  6 : escio:=zcatchall;
603:C      3  otherwise escio(escn);
604:C      3  end; { case }
605:C      2  end;
606:S
607:D      1  procedure bootdam(anyvar f:fib; unum:unitnum; request:damrequesttype);
608:D      2  var
609:D      -2 2  ftype : shortint;
610:D      -4 2  fk : filekind;
611:D      -18 2  dstart : integer;
612:D      -12 2  dsize : integer;
613:D      -16 2  dentry : direntyp;
614:D      -24 2  i,j : integer;
615:S
616:D      2  procedure escesc(ecode:integer);
617:D      2  var
618:D      -4 2  tempec : integer;
619:D      2  begin { convert mini driver escape code to general code }
620:D      2  if ecode<0 then escio(ecode);
621:D      2  if ecode = 1 then iorresult := ord(ibadtitle);
622:D      2  tempec:=ecode mod 1000;
623:D      2  if tempec=80 then escio(2) { no medium }
624:D      2  else if tempec=90 then escio(6){ bad error state }
625:D      2  else escio(4); { read error }
626:D      2  end;
627:S
628:D      2  function minit(msus:msustype):boolean;
629:D      2  var
630:D      -4 2  tempec : integer;
631:D      2  begin
632:D      2  if read then minit:=boot_minit(msus)
633:D      2  else
634:D      2  begin
635:D      2  try
636:D      2  minit:=true;
637:D      2  driverkey := false;
638:D      2  boot_lifhead(dsize,dstart);
639:D      2  driverkey := true;
640:D      2  recover begin
641:D      2  driverkey := true;
642:D      2  if escapecode=-1 then minit:=false
643:D      2  else escio(escapecode);
644:D      2  end;
645:D      2  end;
646:D      2  end;
647:S
648:D      2  function mfopen(anyvar fname:string255; var xaddr,length:integer;
649:D      2  var ftype:shortint):boolean;
650:D      2  var
651:D      -10 2  lfname : lifname;
652:D      -14 2  i : integer;
653:D      -18 2  fk, fkind : filekind;
654:D      2  begin
655:D      2  if read then mfopen:=boot_mfopen(fname,xaddr,length,ftype)
656:D      2  else

```



```

657:C      4      begin
658:C      4      try
659:C      5      mlopen:=true;
660:C      5      if (strlen(fname)=0) or (strlen(fname)>10) then escape(1);
661:C      5      for i:=1 to 10 do
662:C      6      if i>strlen(fname) then tfname[i]:=' ' else tfname[i]:=fname[i];
663:C      5
664:C      5      driverkey := false;
665:C      5      boot_findfile(dentry,tfname,dsize,dstart);
666:C      5      driverkey := true;
667:C      5
668:C      5      fkind:=datafile; ftype:=dentry^.ftype;
669:C      5      for fk:=lastfkind downto untypedfile do
670:C      6      if efttable^[fk]=ftype then fkind:=fk;
671:C      5      if fkind=sysfile then xaddr:=dentry^.extension
672:C      6      else xaddr:=0;
673:C      5      if fkind=datafile then length:=dentry^.extension
674:C      6      else length:=dentry^.fsize*256;
675:C      5      recover
676:C      5      begin
677:C      5      driverkey := true;
678:C      5      mlopen:=false;
679:C      5      if escapecode<>-1 then escesc(escapecode);
680:C      5      end;
681:C      4      end;
682:C      3      end;
683:C      2
684:C      2      procedure mfclose;
685:C      3      begin
686:C      3      if gread then boot_mfclose;
687:C      3      end;
688:C      2
689:C      2      begin { bootdam }
690:C      2      ioresult:=ord(inoerror);
691:C      2      case request of
692:C      3      getvolumename: strptr(addr(f))^ := "BOOT_DEVICE";
693:C      3      openfile:
694:C      4      try
695:C      4      := 0;
696:C      4      for i := 1 to strlen(f.ftitle) do if f.ftitle[i] = '/' then j := i;
697:C      4      if strlen(f.ftitle)-j>tidlength then f.ftid :=
698:C      5      else begin
699:C      5      setstrlen(f.ftid, strlen(f.ftitle)-j);
700:C      5      for i := 1 to strlen(f.ftid) do f.ftid[i] := f.ftitle[j+i];
701:C      5      end;
702:C      4      if minit(boot_msus) then
703:C      5      begin
704:C      5      ftype:=1;
705:C      5      if mlopen(f.ftitle,f.startaddress,f.pEOF,ftype) then
706:C      6      begin
707:C      6      f.fkind:=datafile;
708:C      6      for fk:=lastfkind downto untypedfile do
709:C      7      if efttable^[fk]=ftype then f.fkind:=fk;
710:C      6      if gread then f.fileid := 0
711:C      6      else f.fileid := dentry^.fstart*256;
712:C      6      f.fEOF:=f.pEOF; f.fisnew:=false;
713:C      6      if not f.fbuffered then f.am:=amtable^[UNTYPEDFILE]
714:C      7      else f.am:=amtable^[f.fkind];
715:C      6      end
716:C      6      else ioresult:=ord(inofile);

```

```

717:C      5      end
718:C      5      else with boot_msus do
719:C      6      ioresult:=ord(inounit);
720:C      4      recover ioresult:=ord(escio(escapecode));
721:C      3      closefile:
722:C      3      try
723:C      4      mfclose;
724:C      4      recover ioresult:=ord(escio(escapecode));
725:C      3      otherwise
726:C      3      ioresult:=ord(ibadrequest);
727:C      3      end; { case request }
728:C      2      end; { bootdam }
729:C      2
730:D      1      procedure initbootdam;
731:D      2      type
732:D      2      lrec = packed record
733:D      3      pad : packed array[0..15] of char;
734:D      3      mbptr : ^char; { offset 16 }
735:D      3      mbsize: integer; { offset 20 }
736:D      2      end;
737:D      2      var
738:D      2      boot_space [-300]:^lrec;
739:C      2      begin
740:C      2      if boot_id<3 then gread:=false
741:C      3      else gread:=not boot_flags[7];
742:C      2
743:C      2      if gread then newbytes(boot_space^.mbptr,boot_space^.mbsize );
744:C      2      end; { initboot dam }
745:C      2
746:C      1      procedure boottm(f:fbp; request: amrequesttype;
747:D      2      anyvar buffer: window; bufsize,position: integer);
748:D      2
749:C      2      begin
750:C      2      if gread then
751:C      3      begin
752:C      3      with f^ do
753:C      4      begin
754:C      4      ioresult:=ord(inoerror);
755:C      4      if request=readbytes then
756:C      5      begin
757:C      5      if ((position+bufsize)>f.pEOF) or
758:C      6      (position<0) then ioresult:=ord(znoblock)
759:C      6      else
760:C      6      if (position mod 256)<>0 then ioresult:=ord(zbadmode)
761:C      7      else
762:C      7      begin
763:C      7      try
764:C      8      boot_mread(position div 256,bufsize,buffer,false);
765:C      8      recover ioresult:=ord(escio(escapecode));
766:C      7      end;
767:C      5      end
768:C      5      else ioresult:=ord(zbadmode);
769:C      4      end;{ with }
770:C      3      end
771:C      3      else begin
772:C      3      driverkey := false;
773:C      3      minitio(f.request,buffer,bufsize,position);
774:C      3      driverkey := true;
775:C      3      end;
776:C      2      end; { boottm }

```



```

777:D -53 1
778:D 2 function srmnode(sc: shortint): string2;
779:D 2 const scsize = hex('10000'); nodecom = chr(128+6);
780:D 2 command = hex('604003'); data = hex('604005');
781:D 2 type charptr = ^char;
782:D -4 2 var bnode: string2;
783:D -12 2 node, i: integer;
784:C 2
785:C 2 begin
786:C 3 try
787:C 3 charptr(ord(sc*scsize+command))^ := nodecom;
788:C 3 repeat until charptr(ord(sc*scsize+command))^ = chr(0);
789:C 3 node := ord(charptr(ord(sc*scsize+data))^);
790:C 3 if node < 10 then bnode := '0' else bnode := '';
791:C 3 strwrite(bnode, strlen(bnode)+1, i, node:1);
792:C 3 srmnode := bnode;
793:C 3 recover if escapecode = -12 then srmnode := '' else escape(escapecode);
794:C 2 end;
795:D 1 function bootnode: string2;
796:D 1 const srmtyp = 7*32+1;
797:C 2
798:C 2 begin
799:C 2 bootnode := '';
800:C 4 if mtype = srmtyp then bootnode := srmnode(mscode);
801:C 2 end;
802:S
803:C 1 end { module }
804:S
805:S
806:C 2 $include 'BOOTDAM'S';
806:D 1 $include 'LOADER' $;
807:D 1 module loader;
808:S
809:D 1 import sysglobals, asm;
810:S
811:D 1 export
812:S
813:D 1 const blocksize = fblksize;
814:D 1 vlength = 7;
815:D 1 flength = 15;
816:D 1
817:D 1 type
818:S
819:D 1 volname = string[vlength];
820:D 1 filename = string[flength];
821:S
822:D 1 dirrange = 0..maxint; {0..MAXlongDIR; }
823:S
824:D 1 (* the following declaration serves for a "library directory" *)
825:S
826:D 1 direntry = record
827:D 1 dfirstblk: shortint; {module starting block*}
828:D 1 dlastblk: shortint; {block following end*}
829:D 1 {NOTE: for DIR[0], these refer to the library directory itself}
830:D 1 case dfkind: filekind of
831:D 1 untypedfile: {library info in DIR[0]*}
832:D 1 (dvid: volname; {name of library*}
833:D 1 deovblk: shortint; {block following library*}
834:D 1 dnumfiles: dirrange; {num modules in library*}
835:D 1 dloadtime: shortint; {time of last modification*}

```

```

836:D 1 dlastboot: daterec; {most recent date setting*}
837:D 1
838:D 1 datafile..lastfkind: {title of module*}
839:D 1 (dtid: filename; {title of module*}
840:D 1 dlastbyte: 1..fblksize; {1..256 bytes in last block*}
841:D 1 daccess: daterec; {last modification date*}
842:D 1 end (*direntry*);
843:S
844:D 1 ubyterec = packed record ub: byte end;
845:D 1 sbyterec = packed record sb: -128..127 end;
846:D 1 word = 0..65535;
847:D 1 wordrec = packed record w: word end;
848:D 1 wordrecptr = ^wordrec;
849:D 1 wordlist = packed array[0..maxint] of word;
850:D 1 wordlistptr = ^wordlist;
851:S
852:D 1 symboltable = array[0..maxint] of char;
853:D 1 symtableptr = ^symboltable;
854:S
855:D 1 symbol = string[255];
856:D 1 symbolptr = ^symbol;
857:S
858:D 1 datatype = (sbyte, sword, sint, fltp, ubyte, uword);
859:D 1 reloctype = (absolute, relocatable, global, general);
860:S
861:D 1 referenceptr = packed record {one or more present if type = general}
862:D 1 case integer of
863:D 1 0: {
864:D 1 adr: 0..16383; {word address of external symbol}
865:D 1 op: (addit, subit); {add or subtract the modifying value}
866:D 1 last: boolean; {indicates end of list}
867:D 1
868:D 1 1:(w: word); {for comparisons}
869:D 1 end;
870:S
871:D 1 gvrptr = ^generalvalue;
872:D 1 veptr = ^valueextension;
873:D 1 textdesptr = ^textdescriptor;
874:D 1 fileptr = ^phyle;
875:D 1 moddesptr = ^moduledescriptor;
876:D 1 refptrptr = ^referenceptr;
877:D 1 sortlistptr = ^sortlist;
878:D 1 moddirptr = ^moduledirectory;
879:D 1 filedirptr = ^filedirectory;
880:D 1 ptrtableptr = ^ptrtable;
881:D 1 patchdesptr = ^patchdescriptor;
882:S
883:S
884:D 1 addrec = record case integer of {a universal address}
885:D 1 -1: (rp: referenceptr); {a two byte object; rest are 4 bytes;}
886:S
887:D 1 1: (i: integer); {to change an address to integer}
888:D 1 2: (a: integer); {to do address arithmetic}
889:D 1 3: (p: ^integer); {to dereference}
890:S
891:D 1 4: (sb: ^sbyterec); {signed byte record}
892:D 1 5: (sw: ^sword); {signed word}
893:D 1 6: (si: ^sint); {signed integer}
894:D 1 7: (fp: ^fltp); {floating point}
895:D 1 8: (ub: ^ubyte); {unsigned byte record}

```



```

896:D 1 9: (uw: wordrecptr); {^unsigned word record}
897:S
898:D 1 10: (gvp: gvrptr); {^generalvalue}
899:D 1 11: (vep: veptr); {^valueextension}
900:D 1 12: (tdp: textdescptr); {^textdescriptor}
901:D 1 13: (syp: symbolptr); {^symbol (string) }
902:D 1 14: (stp: symtableptr); {^symboltable}
903:D 1 15: (wlp: wordlistptr); {^wordlist}
904:D 1 16: (php: fileptr); {^phyle}
905:D 1 17: (mdp: moddescptr); {^moduledescriptor}
906:D 1 18: (rpp: refptr); {^referenceptr}
907:D 1 19: (sdp: ^sortdesc); {^sortlist}
908:D 1 20: (slp: sortlistptr); {^sortlist}
909:D 1 21: (drp: moddirptr); {^moduledirectory}
910:D 1 22: (fdp: filedirptr); {^filedirectory}
911:D 1 23: (bmp: ^bitmap); {^ptrtable}
912:D 1 24: (fbp: ^fbp); {^ptrtable}
913:D 1 25: (ptp: ptrtableptr); {^ptrtable}
914:D 1 26: (ilp: indexlist); {^ptrtable}
915:D 1 27: (pdp: patchdescptr); {^ptrtable}
916:D 1 28: (cp: ^char); {^ptrtable}
917:D 1 29: (arp: ^addrec); {^ptrtable}
918:D 1 end;
919:S
920:D 1 generalvalue = packed record
921:D 1 primarytype: reloctype; {allows quick indication of most common types}
922:D 1 datasize: datatype; {specifies 1, 2, 4 or 8 bytes, signed or not}
923:D 1 patchable, {specifies self relative field in branch}
924:D 1 valueextend: boolean; {indicates the presence of valueextension}
925:D 1 case longoffset: boolean of {1 or 3 byte offset}
926:D 1 false: (short: byte); {unsigned 8 bits}
927:D 1 true: (long: ^long); {unsigned 24 bit value}
928:D 1 end;
929:S
930:D 1 valueextension = packed record {present if valueextend bit above is set}
931:D 1 case datatype of
932:D 1 sbyte, sword, sint, {value: integer};
933:D 1 ubyte, uword: {value: real};
934:D 1 fltptr: {value: real};
935:D 1 end;
936:S
937:D 1 phyle = file of char;
938:S
939:S
940:D 1 moduledirectory = packed record
941:D 1 date: daterec; {date of creation}
942:D 1 revision: daterec; {producer's revision date number}
943:D 1 producer: char; {A = assembler, C = compiler, L = linker, etc.}
944:D 1 systemid: byte; {system version number (hard or soft, etc.)}
945:D 1 notice: string[80]; {space for whatever comments may be desired}
946:D 1 directorysize: integer; {size of module directory, in bytes}
947:D 1 modulesize: integer; {total size of module, in bytes}
948:D 1 executable: boolean; {module is executable, has start address}
949:D 1 relocatable: integer; {number of relocatable bytes requested}
950:D 1 relocatablebase: integer; {current origin of relocatable code}
951:D 1 globalsize: integer; {number of global bytes requested}
952:D 1 globalbase: integer; {A5 relative origin of global area}
953:D 1 globalbase: integer; {A5 relative origin of global area}
954:D 1 globalbase: integer; {A5 relative origin of global area}
955:D 1 extblock, {module relative block of EXT table}

```

```

956:D 1 extsize, {size of EXT table, in bytes}
957:D 1 defblock, {module relative block of DEF table}
958:D 1 defsize, {size of DEF table, in bytes}
959:D 1 sourceblock, {module relative block of DEFINE SOURCE}
960:D 1 sourcesize, {size of source, in bytes}
961:S
962:D 1 textrecords: integer; {number of TEXT records}
963:S
964:S {Remainder of directory is made up of variable length
965:S records. Strings begin and end on word (even byte)
966:S boundaries. The directory itself may cross block
967:S boundaries. General value or address records
968:S (GVR's, see description later)
969:S occurring below have the short variant offset; the
970:S offset itself is the length of the GVR to assist in
971:S stepping quickly through the list.}
972:S
973:S
974:D 1 {mname: string[ (variable) ]; {name of module}
975:S
976:D 1 {startaddress: gvr; {execution address, present only
977:D 1 if executable}
978:S
979:D 1 {repeat for each text record {list of TEXT records}
980:D 1 {textstart, {module relative block of TEXT record}
981:D 1 {textsize, {size of TEXT record, in bytes}
982:D 1 {refstart, {module relative block of REF table}
983:D 1 {refsize: integer; {size of REF table, in bytes}
984:D 1 {loadaddress: gvr; {location to load the TEXT}
985:D 1 {end}
986:D 1 end;
987:S
988:D 1 textdescriptor = record
989:D 1 textstart, {module relative block of TEXT record}
990:D 1 textsize, {size of TEXT record, in bytes}
991:D 1 refstart, {module relative block of REF table}
992:D 1 refsize: integer; {size of REF table, in bytes}
993:D 1 end;
994:S
995:D 1 bitmap = packed array[0..maxint] of boolean;
996:S
997:D 1 patchdescriptor = record
998:D 1 patchlist: patchdescptr; {head of list of patch descriptors}
999:D 1 patchref: addrec;
1000:D 1 end;
1001:S
1002:D 1 moduledescriptor = record
1003:D 1 link: moddescptr; {descriptors will be chained together}
1004:S
1005:D 1 case patchmod: boolean of
1006:D 1 true: {
1007:D 1 patchlink: moddescptr; {patchmods are additionally linked}
1008:D 1 patchlist: patchdescptr; {head of list of patch descriptors}
1009:D 1 patchbase: integer; {relocatable address of patch space}
1010:D 1 patchsize: integer; {total bytes of patch space}
1011:S
1012:D 1 false: {
1013:D 1 defaddr: addrec; {address of DEF table}
1014:D 1 defsize: integer; {size of DEF table, in bytes}
1015:S

```



```

1016:D 1 case resolved: boolean of           {only present during loading}
1017:D 1 true: (startaddr: integer;
1018:D 1   progname: tid;
1019:D 1   ucase, lastmodule: boolean);
1020:D 1 false: (extaddr:   address;           {address of EXT table}
1021:D 1   listaddr:   wordlistptr;           {index pointers into EXT table}
1022:D 1   listsize:   shortint;             {number of entries in list}
1023:D 1   unresbits:   address;             {flag to indicate unresolved symbols}
1024:D 1   relocbase,   integer;             {relocatable address of module}
1025:D 1   globase:      integer;             {global base address}
1026:D 1   relocdelta,   integer;             {deltas for relocation}
1027:D 1   globaldelta:   integer;             {deltas for relocation}
1028:D 1   filefib:      address;             {file info block for module}
1029:D 1   fileblock:    shortint;           {file relative block of module}
1030:D 1   directory:   address;             {memory for processing modules}
1031:D 1 end;
1032:S
1033:D 1 filedirectory = array[0..maxint] of direntry;
1034:S
1035:D 1 sortdesc = record
1036:D 1   modsp: moddescptr;
1037:D 1   case integer of
1038:D 1     1: (ext: symbolptr; n: shortint);
1039:D 1     2: (def: address);
1040:D 1   end;
1041:D 1 sortlist = array[0..maxint] of sortdesc;
1042:S
1043:D 1 indexlist = array[1..maxint] of shortint;
1044:S
1045:D 1 ptrtable = array[0..maxint] of address;
1046:S
1047:D 1 var {memory management: }
1048:D 1   a5['g_dollar']: integer;
1049:D 1   lowheap,         integer;           {next available memory}
1050:D 1   highheap:        address;          {last available memory}
1051:S
1052:D 1   {input file information: }
1053:D 1   fdirectory:       filedirptr;      {old library directory pointer}
1054:D 1   loadfib:          address;         {pointer to FIB for open file}
1055:D 1   wrongbyte:        integer;         {error information}
1056:D 1   linkmodname:     address;         {ptr to name of module being linked}
1057:S
1058:D 1   {linker information: }
1059:D 1   newmods:          moddescptr;      {modules currently being processed}
1060:D 1   allresolved:      boolean;         {flag indicating whether any ext's}
1061:D 1   totalreloc:        integer;        {total bytes of areas}
1062:D 1   totalglobal:      integer;        {total bytes of areas}
1063:D 1   startreloc:       integer;        {starting addresses for areas}
1064:D 1   startglobal:      integer;        {starting addresses for areas}
1065:S
1066:D 1   ehheap: anyptr;      {HEAP MARK FOR MEM MANAGING*}
1067:D 1   edefs: moddescptr;
1068:D 1   userstack: integer;
1069:D 1   entrypoint: moddescptr;
1070:D 1   eglobal: integer;    {*PERMANENT BASE OF DATA AREA*}
1071:D 1   sysdefs: moddescptr; {list of permanent module descriptors}
1072:S
1073:D 1   endmod: moddescptr; {marks end of list of new modules}
1074:D 1   libfound: boolean;
1075:D 1   wrongrec: integer;   {error information}

```

```

1076:D -80 1 startdefs;
1077:D -88 1 totaldefs: integer;           {DEF table information}
1078:S
1079:D -88 1 procedure markuser;
1080:D -88 1 procedure releaseuser;
1081:S
1082:D -88 1 procedure loadinfo(modnum: shortint; all, resolveexts: boolean);
1083:D -88 1 procedure openlink(extra: address); {formerly 'openlinkfile'}
1084:D -88 1 procedure getbytes(var p: integer; size: integer);
1085:D -88 1 procedure checkrev;
1086:D -88 1 procedure matchfile;
1087:D -88 1 procedure countcode;
1088:D -88 1 procedure loadtext(onheap: boolean);
1089:D -88 1 procedure zeromem(start: anyptr; size: integer);
1090:D -88 1 procedure closefiles;
1091:D -88 1 procedure movedefs(newstartdefs: integer);
1092:S
1093:D -88 1 procedure loadq(filetogo: fid);    {name of file to be loaded}
1094:D -88 1 procedure initloader;
1095:S
1096:D -88 1 implement
1097:S
1098:D -88 1 type pointer = ^integer;
1099:D -88 1 address = integer;
1100:S
1101:D -88 1 var sysdeftable['sysdeftable']:
1102:D -88 1   moduledescptr; {initial ROM symbol table}
1103:S
1104:D 1 procedure evalgvr(var gvalue, gvptr: integer;
1105:D 1   modptr: moddescptr); external;
1106:S
1107:D 1 procedure relocate(reftop, reindex: address;
1108:D 1   var object: address;
1109:D -88 2   modptr: moddescptr); external;
1110:S
1111:D 1 procedure getbytes(var p: integer; size: integer);
1112:D 2 begin
1113:D 2   if odd(size) then size := size + 1;
1114:D 2   p := lowheap.a; lowheap.a := lowheap.a + size;
1115:D 2   if lowheap.a > highheap.a then escape(112);
1116:D 2 end;
1117:S
1118:D 1 procedure readblocks(anyvar f: fib; anyvar obj: integer; size, block: integer);
1119:D 2 begin
1120:D 2   call (f.am, addr(f), readbytes, obj, size, block*fbksize);
1121:D 2   if ioresult <> ord(inoerror) then escape(-10);
1122:D 2 end;
1123:S
1124:D 1 procedure markuser;
1125:D 2 begin
1126:D 2   entrypoint := nil; mark(eheap);
1127:D 2   eglobal := userstack; edefs := sysdefs;
1128:D 2 end;
1129:S
1130:D 1 procedure releaseuser;
1131:D 2 begin
1132:D 2   entrypoint := nil; release(eheap);
1133:D 2   userstack := eglobal; sysdefs := edefs;
1134:D 2 end;
1135:S

```



```

1136:D 1 procedure openlinkf(extra: addrec);
1137:D -4 2 var directsize: integer;
1138:D -8 2 slop: integer;
1139:C 2 begin if ioresult = 0 then
1140:C 3 begin
1141:C 3 if loadfib.fbp^.fkind <> codefile then escape(116);
1142:C 3 highheap.a := highheap.a - blocksize;
1143:C 3 if highheap.a < lowheap.a then escape(112);
1144:C 3 fdirectory := highheap.fdp;
1145:C 3 readblocks(loadfib.php^,fdirectory^,blocksize,0);
1146:C 3 directsize := (fdirectory[0].dnumfiles + 1)*sizeof(direntry);
1147:C 3 slop := (-directsize) mod blocksize;
1148:C 3 highheap.a := highheap.a + blocksize - directsize - slop;
1149:C 3 if directsize > blocksize then
1150:C 4 begin
1151:C 4 if highheap.a < lowheap.a then escape(112);
1152:C 4 moveleft(fdirectory^, highheap.fdp^, blocksize);
1153:C 4 extra.a := highheap.a + blocksize;
1154:C 4 readblocks(loadfib.php^,extra.p^,directsize - blocksize,1);
1155:C 4 end;
1156:C 3 if slop > 0 then
1157:C 4 begin
1158:C 4 extra := highheap; highheap.a := highheap.a + slop;
1159:C 4 moveright(extra.fdp^, highheap.fdp^, directsize);
1160:C 4 end;
1161:C 3 fdirectory := highheap.fdp;
1162:C 3 end;
1163:C 3 else begin fdirectory := nil; (directory invalid)
1164:C 3 loadfib := extra.arp^; (restore chain of FIB's)
1165:C 3 lowheap := extra; (zap unopened FIB)
1166:C 3 end;
1167:C 2 end; {openlinkfile}
1168:S
1169:D 1 procedure matchdefext(
1170:D 2 var resolved, matched: boolean;
1171:D 2 matchflag: byte;
1172:D 2 deftable, exttable: symtableptr;
1173:D 2 extlist: wordlistptr;
1174:D 2 listlength: shortint;
1175:D 2 deftablelength: integer);
1176:D -88 2 external;
1177:S
1178:D 1 procedure matchl(modptr: moddescptr);
1179:D -5 2 var defmodptr: moddescptr; matched: boolean;
1180:C 2 begin with modptr do
1181:C 3 begin
1182:C 3 defmodptr := link;
1183:C 3 while not resolved and (defmodptr <> nil) do
1184:C 4 begin
1185:C 4 if not defmodptr^.patchmod then
1186:C 5 matchdefext(resolved, matched, 0,
1187:C 5 defmodptr^.defaddr.stp, extaddr.stp,
1188:C 5 listaddr, listsize, defmodptr^.defsize);
1189:C 4 defmodptr := defmodptr^.link;
1190:C 4 end;
1191:C 3 allresolved := allresolved and resolved;
1192:C 3 end;
1193:C 2 end;
1194:S
1195:D 1 procedure matchfile;

```

```

1196:D -4 2 var modptr: moddescptr;
1197:C 2 begin
1198:C 2 modptr := newmods;
1199:C 2 while modptr <> nil do
1200:C 3 begin
1201:C 3 matchl(modptr);
1202:C 3 modptr := modptr^.link;
1203:C 3 end;
1204:C 2 end;
1205:S
1206:D 1 procedure loadinfo(modnum: shortint; all, resolveexts: boolean);
1207:D -4 2 var modptr: moddescptr;
1208:D -5 2 matched: boolean;
1209:D -10 2 leftover: address;
1210:D -12 2 modblock: shortint;
1211:S
1212:D 2 procedure alphalist(
1213:D 3 symtable: symtableptr;
1214:D 3 list: wordlistptr;
1215:D 3 listlength: shortint);
1216:D -12 3 external;
1217:S
1218:D 2 procedure makelist(tableptr: symtableptr; (DEF or EXT table)
1219:D 3 length: integer; (length of symbol table)
1220:D 3 bound: shortint; (boundary condition)
1221:D 3 var listptr: wordlistptr; (address of list)
1222:D 3 var listlength: integer); (number of symbols)
1223:S
1224:D -8 3 var i, l: integer;
1225:D -12 3 n, len: shortint;
1226:D -16 3 ptr: addrec;
1227:S
1228:C 3 begin
1229:C 3 n := 0; listptr := lowheap.wlp;
1230:C 3 l := length; i := 0;
1231:S
1232:C 3 while l > 0 do
1233:C 4 begin
1234:C 4 n := n + 1;
1235:C 4 getbytes(ptr.a, sizeof(wordrec)); ptr.uw.w := 1;
1236:C 4 len := ord(tableptr[i]);
1237:C 4 len := len + bound - (len mod bound);
1238:C 4 if bound = 2 (DEF table) then len := len + ord(tableptr[i+len+1]);
1239:C 4 i := i + len;
1240:C 4 l := l - len;
1241:C 4 end;
1242:C 3 listlength := n;
1243:C 3 alphalist(tableptr, listptr, n);
1244:C 3 end; {makelist}
1245:S
1246:D 2 procedure getdeftable;
1247:S
1248:D -4 3 var defptr: addrec;
1249:D -4 3 symptr1: addrec;
1250:D -12 3 symptr2: addrec;
1251:D -20 3 l: integer;
1252:D -24 3 list: wordlistptr;
1253:D -28 3 size: integer;
1254:S

```



```

1255:C      3      begin
1256:C      3      with modptr^ do
1257:C      4      begin
1258:C      4      defsize := directory.drp^.defsize;
1259:C      4      getbytes(defaddr.a, defsize);
1260:C      4      if defsize > 0 then
1261:C      5      begin
1262:C      5      getbytes(defptr.a, defsize);
1263:C      5      readblocks(loadfib.php^, (unitread(funit, ... ioccheck);
1264:C      5      defptr.p^, defsize, modblock + directory.drp^.defblock);
1265:C      5      makelist(defptr.stp, defsize, 2, list, size);
1266:C      5      symptr2 := defaddr;
1267:C      5      for i := 0 to size-1 do
1268:C      6      begin
1269:C      6      symptr1.a := defptr.a+list[i];
1270:C      6      len := strlen(symptr1.syp^);
1271:C      6      len := len + 2 - ord(odd(len));
1272:C      6      len := len + gvrptr(symptr1.a+len)^.short;
1273:C      6      fastmove(symptr1.p, symptr2.p, len);
1274:C      6      symptr2.a := symptr2.a+len;
1275:C      6      end;
1276:C      5      lowheap := defptr;
1277:C      5      end;
1278:C      4      end;
1279:C      3      end;
1280:S
1281:D      2      procedure getexttable;
1282:D      3      var size: integer;
1283:D      3      begin with modptr^, directory.drp^ do
1284:D      4      begin
1285:D      4      if extsize < 8 then extsize := 8;
1286:D      4      getbytes(extaddr.a, extsize);
1287:D      4      if extsize > 8 then
1288:D      5      readblocks(loadfib.php^, extaddr.p^, extsize, modblock + extblock);
1289:D      4      extaddr.stp[0] := chr(0);
1290:D      4      extaddr.stp[4] := chr(0);
1291:D      4      makelist(extaddr.stp, extsize, 4, listaddr, size);
1292:D      4      listsize := size;
1293:D      4      listaddr[0] := 0;
1294:D      4      listaddr[i] := 0;
1295:D      4      end;
1296:D      3      end;
1297:D      2      end;
1298:D
1299:D
1300:D      2      procedure match2;
1301:D      3      var extmodptr: moddescptr;
1302:D      3      begin
1303:D      3      extmodptr := newmods;
1304:D      3      while extmodptr <> nil do with extmodptr^ do
1305:D      4      begin
1306:D      4      if not patchmod then if not resolved then
1307:D      5      begin
1308:D      5      matchdefext(resolved, matched, 0,
1309:D      5      modptr^.defaddr.stp, extaddr.stp,
1310:D      5      listaddr, listsize, modptr^.defsize);
1311:D      5      allresolved := allresolved and resolved;
1312:D      5      end;
1313:D      4      extmodptr := link;
1314:D      3      end;

```

```

1315:C      3      end;
1316:S
1317:C      2      begin {loadinfo}
1318:C      2      modblock := (fblock +) fdirectory^[modnum].dfirstblk;
1319:C      2      modptr := lowheap.mdp;
1320:C      2      getbytes(leftover, sizeof(moduledescrptor, false, false));
1321:C      2      modptr^.directory := lowheap;
1322:C      2      getbytes(leftover, blocksize);
1323:C      2      with modptr^, directory.drp^ do
1324:C      3      begin
1325:C      3      readblocks(loadfib.php^, directory.drp^, blocksize, modblock);
1326:C      3      if directorysize > blocksize then
1327:C      4      begin
1328:C      4      getbytes(leftover, directorysize-blocksize);
1329:C      4      readblocks(loadfib.php^, (unitread(funit ... ioccheck)
1330:C      4      pointer(leftover)^, directorysize-blocksize, modblock+1);
1331:C      4      end
1332:C      3      else lowheap.a := directory.a+directorysize;
1333:C      3      getdeftable;
1334:C      3      allresolved := true;      matched := false;
1335:C      3      match2;
1336:C      3      if matched or all then
1337:C      4      begin
1338:C      4      libfound := true;
1339:C      4      link := newmods;      newmods := modptr;
1340:C      4      patchmod := false;
1341:C      4      resolved := false;      {set appropriate variant}
1342:C      4      filefib := loadfib;      {remember mass memory location for later}
1343:C      4      fileblock := modblock;
1344:C      4      end
1345:C      3      getexttable;
1346:C      3      matchdefext(resolved, matched,
1347:C      3      0(patchable bit *** fix later! **),
1348:C      3      defaddr.stp, extaddr.stp, listaddr,
1349:C      3      listsize, defsize);
1350:C      3      if resolveexts then match1(modptr);
1351:C      3      end
1352:C      3      else lowheap.mdp := modptr;
1353:C      3      end;
1354:S
1355:C      2      end; {loadinfo}
1356:S
1357:D      1      procedure reversepointers;
1358:D      2      var modptr, lastptr, nextptr: moddescptr;
1359:D      2      begin
1360:D      2      modptr := newmods; lastptr := endmod;
1361:D      2      while modptr <> endmod do with modptr^ do
1362:D      3      begin nextptr := link;      link := lastptr;
1363:D      3      lastptr := modptr;      modptr := nextptr;
1364:D      3      end;
1365:D      2      newmods := lastptr;
1366:D      2      end;
1367:S
1368:D      1      procedure countcode;
1369:D      2      var modptr: moddescptr;
1370:D      2      begin
1371:D      2      reversepointers;
1372:D      2      modptr := newmods;
1373:D      2      totalreloc := 0; totalglobal := 0;
1374:D      2      while modptr <> endmod do with modptr^ do

```



```

1375:C      4      begin
1376:C      4      with directory.drp^ do
1377:C      5      begin
1378:C      5      totalreloc := totalreloc + relocatablesize + ord(odd(relocatablesize));
1379:C      5      totalglobal := totalglobal + globalsize + ord(odd(globalsize));
1380:C      5      end;
1381:C      4      modptr := link;
1382:C      4      end;
1383:C      2      end;
1384:S
1385:D      1      procedure closefiles;
1386:C      2      begin
1387:C      2      while loadfib.php <> nil do
1388:C      3      begin
1389:C      3      (close(loadfib.php^));
1390:C      3      with loadfib.fbp^, unitable^[funit] do
1391:C      4      call(dam, loadfib.fbp^, funit, closefile);
1392:C      4      loadfib.a := loadfib.a - sizeof(addrrec);
1393:C      3      loadfib := loadfib.arp^;
1394:C      3      end;
1395:C      2      end;
1396:S
1397:D      1      procedure resolve;
1398:D      -4      2      var modptr: moddescptr;
1399:D      -12      2      mbase, mgbase: integer;
1400:D      -14      2      len: shortint;
1401:D      -16      2      i: shortint;
1402:S
1403:C      2      begin
1404:C      2      modptr := newmods;
1405:C      2      mbase := startreloc; mgbase := startglobal;
1406:C      2      while modptr <> endmod do with modptr^ do
1407:C      4      begin
1408:C      4      with directory.drp^ do
1409:C      5      begin
1410:C      5      relocbase := mbase; relocdelta := mbase - relocatablebase;
1411:C      5      mbase := mbase + relocatablesize + ord(odd(relocatablesize));
1412:C      5      gbase := mgbase; globaldelta := mgbase - globalbase;
1413:C      5      mgbase := mgbase - globalsize - ord(odd(globalsize));
1414:C      5      end;
1415:C      4      modptr := link;
1416:C      4      end;
1417:C      2      end; {resolve}
1418:S
1419:D      1      procedure createdefs;
1420:D      -4      2      var modptr: moddescptr;
1421:D      -4      2      newmodptr;
1422:D      -24      2      sp, ptr1, ptr2, enddefs: addrrec;
1423:D      -26      2      len: shortint;
1424:C      2      begin
1425:C      2      reversepointers;
1426:C      2      entrypoint := nil;
1427:C      2      modptr := newmods;
1428:C      2      startdefs := lowheap.a;
1429:C      2      while modptr <> endmod do with modptr^ do
1430:C      4      begin
1431:C      4      ptr1 := defaddr;
1432:C      4      enddefs.a := ptr1.a + defsize;
1433:C      4      getbytes(newmodptr.a, sizeof(moduledescriptor, false, true));
1434:C      4      with newmodptr.mdp^ do

```

```

1435:C      5      begin
1436:C      5      patchmod := false; resolved := true;
1437:C      5      progname := modptr^.filefib.fbp^.ftid {togo};
1438:C      5      ucase := unitable^[modptr^.filefib.fbp^.funit].uppercase;
1439:C      5      if modptr^.directory.drp^.executable then
1440:C      6      begin
1441:C      6      sp.a := modptr^.directory.a + sizeof(moduledirectory);
1442:C      6      sp.a := sp.a + strlen(sp.syp^)*2 - ord(odd(strlen(sp.syp^)));
1443:C      6      evalgvr(startaddr, sp.i, modptr);
1444:C      6      lastmodule := entrypoint=nil;
1445:C      6      entrypoint = newmodptr.mdp; {moddescptr(newmodptr.a - loaddelta);}
1446:C      6      end;
1447:C      6      else begin startaddr := 0;
1448:C      6      lastmodule := false;
1449:C      6      end;
1450:C      5      link := sysdefs;
1451:C      5      sysdefs := newmodptr.mdp; {moddescptr(newmodptr.a - loaddelta);}
1452:C      5      defaddr.a := lowheap.a {- loaddelta};
1453:C      5      while ptr1.a < enddefs.a do
1454:C      6      begin
1455:C      6      len := strlen(ptr1.syp^)+2 - ord(odd(strlen(ptr1.syp^)));
1456:C      6      getbytes(ptr2.a, len); fastmove(ptr1.p, ptr2.p, len);
1457:C      6      ptr1 := ptr1.a + len;
1458:C      6      getbytes(ptr2.a, sizeof(generalvalue, false));
1459:C      6      with ptr2.gvp^ do
1460:C      7      begin primarytype := absolute; datasize := sint;
1461:C      7      patchable := false; valueextend := true;
1462:C      7      longoffset := false; short := 6;
1463:C      7      end;
1464:C      6      getbytes(ptr2.a, sizeof(valueextension, sint));
1465:C      6      evalgvr(ptr2.vgp^.value, ptr1.i, modptr);
1466:C      6      end;
1467:C      5      defsize := lowheap.a - (defaddr.a + loaddelta);
1468:C      5      end;
1469:C      4      modptr := link;
1470:C      4      end;
1471:C      2      totaldefs := lowheap.a - startdefs;
1472:C      2      end;
1473:S
1474:D      1      procedure movedefs(newstartdefs: integer);
1475:D      -4      2      var modptr: moddescptr;
1476:D      -8      2      defdelta: integer;
1477:D      -12      2      previous: ^addrrec;
1478:C      2      begin
1479:C      2      defdelta := newstartdefs - startdefs;
1480:C      2      previous := addr(sysdefs);
1481:C      2      modptr := sysdefs;
1482:C      2      while modptr <> endmod do with modptr^ do
1483:C      4      begin
1484:C      4      previous^.a := previous^.a + defdelta;
1485:C      4      if entrypoint = modptr then entrypoint := previous^.mdp;
1486:C      4      defaddr.a := defaddr.a + defdelta;
1487:C      4      previous := addr(link);
1488:C      4      modptr := link;
1489:C      4      end;
1490:C      2      fastmove(pointer(startdefs), pointer(newstartdefs), totaldefs);
1491:C      2      startdefs := newstartdefs;
1492:C      2      end;
1493:D      -88      1
1494:D      1      procedure loadtext(onheap: boolean);

```



```

1495:S      2 const maxrefsize = 254;
1496:D
1497:S
1498:D      -4 2 var modptr:      moddescptr;      {current module being loaded}
1499:S
1500:D      -4 2      textbuffer,      {base of text record buffer}
1501:D      -4 2      object,object0,      {object in text record being modified by ref record}
1502:D      -4 2      refbuffer,      {base of ref table buffer}
1503:D      -4 2      oldindex:      {pointer to old text descriptors}
1504:D      -24 2      address;
1505:S
1506:D      -24 2      loaddelta,      {loader displacement}
1507:D      -24 2      oldtextrec:      {text records left to process from old module}
1508:D      -32 2      integer;
1509:S
1510:S
1511:S
1512:C      2 begin {procedure loadtext}
1513:C      2   if onheap then
1514:C      3     begin
1515:C      4       getbytes(textbuffer.a, totalreloc (+ blocksize));
1516:C      4       loaddelta := textbuffer.a - startreloc;
1517:C      3     end
1518:C      3   else loaddelta := 0;
1519:C      2   resolve;
1520:C      2   modptr := newmods;
1521:C      2   while modptr <> endmod do with modptr^, directory.drp^ do
1522:C      4     begin
1523:C      4       linkmodname.a := directory.a + sizeof(moduledirectory);
1524:C      4       oldindex.a := linkmodname.a + strlen(linkmodname.syp^)+ 2 -
1525:C      4         ord(odd(strlen(linkmodname.syp^)));
1526:C      4       if executable then oldindex.a := oldindex.a + oldindex.gvp^.short;
1527:C      4       for oldtextrec := 1 to textrecords do
1528:C      5         with oldindex.tdp^ do
1529:C      6           begin
1530:C      6             oldindex.a := oldindex.a + sizeof(textdescriptor);
1531:C      6             evalgr(object.i, oldindex.i, modptr);
1532:C      6             if textsize > 0 then
1533:C      7               begin
1534:C      7                 if object.a >= startreloc
1535:C      8                   then if object.a < startreloc + totalreloc
1536:C      8                     then object.a := object.a + loaddelta;
1537:C      7               end
1538:C      7               readblocks(modptr^.filefib.php^, {unitread(modptr^.fileunit,...iocheck)
1539:C      7                 object.p^,textsize, fileblock + textstart});
1540:C      7             if refsize > 0 then
1541:C      8               begin
1542:C      8                 getbytes(refbuffer.a, refsize);
1543:C      8                 readblocks(modptr^.filefib.php^,
1544:C      8                   refbuffer.p^,refsize, fileblock + refstart);
1545:C      8                 object0 := object;
1546:C      8                 try relocate(lowheap.a, refbuffer.a, object.a, modptr);
1547:C      9                 recover
1548:C      9                   begin
1549:C      9                     wrongbyte := object.a-object0.a;
1550:C      9                     wrongrec := oldtextrec;
1551:C      9                     escape(escapecode);
1552:C      9                   end;
1553:C      9                 lowheap := refbuffer;
1554:C      9                 end;

```

```

1555:C      7       end;
1556:C      6       end; {for oldtextrec}
1557:C      4       modptr := link;
1558:C      4       end;
1559:S
1560:C      2   createdefs;
1561:C      2   closefiles;
1562:S
1563:C      2   if onheap then fastmove(textbuffer.p, pointer(startreloc), totalreloc);
1564:S
1565:C      2 end;
1566:S
1567:D      -8 1 procedure zeromem(start: anyptr; size: integer);
1568:D      2 var ptr, endp: address;
1569:C      2 begin
1570:C      3   ptr.p := start; endp.a := ptr.a + size;
1571:C      3   while ptr.a < endp.a do
1572:C      3     begin
1573:C      3       ptr.sw^ := 0;
1574:C      3       ptr.a := ptr.a + 2;
1575:C      3     end;
1576:C      2 end;
1577:S
1578:D      1 procedure checkrev;
1579:C      2 begin
1580:C      2   if newmods^.directory.drp^.systemid <> 3 then escape(118);
1581:C      2 end;
1582:S
1583:D      -122 1 procedure loadq(filetoq:fid);
1584:S
1585:D      -122 2 type str9 = packed array[1..9] of char;
1586:S
1587:D      -122 2 const loading = str9['Loading '];
1588:S
1589:D      -126 2 var modnum, esc: shortint;
1590:D      -130 2 extra: address;
1591:D      -138 2 i, ior: integer;
1592:S
1593:C      2 begin
1594:S
1595:C      2 loadfib.php := nil;
1596:C      2 try
1597:C      3   releaseuser;      {get rid of last program}
1598:C      3   mark(lowheap.p);      highheap.a := userstack;
1599:C      3   startreloc := lowheap.a;
1600:C      3   newmods := sysdefs;      endmod := sysdefs;
1601:S
1602:C      3   getbytes(extra.a, sizeof(address)); {save old value of loadfib}
1603:C      3   extra.arp^ := loadfib;
1604:C      3   getbytes(loadfib.a, sizeof(fib,1)); {create a FIB for new file}
1605:C      3   with loadfib.fbp^, unitable^ [sysunit] do
1606:C      4     begin
1607:C      4       fbuffered := false;
1608:C      4       ftitle := filetoq;
1609:C      4       funit := sysunit;
1610:C      4       call(dam, loadfib.fbp^, sysunit, openfile);
1611:C      4       am := tm;
1612:C      4       end;
1613:C      3   openlinkf(extra);
1614:S

```



```

1615:C      3      1 := strlen(filetogo)+1; setstrlen(filetogo, i+9);
1616:C      3      filetogo[i] := ' ';
1617:C      3      moveright(filetogo[i],      filetogo[i0], i);
***WARNING: (line 1618): 'R00R' of a constant may not be supported on other implementations
1618:C      3      fastmove(addr(loading),      addr(filetogo[i]), 9);
1619:C      3      cpymsg(filetogo);
1620:C
1621:C      3      if fdirectory = nil then escape(-10);
1622:C      3      for modnum := 1 to fdirectory[0].dnumfiles do
1623:C      4          begin loadinfo(modnum, true, false); checkrev; end;
1624:C      3      allresolved := true; matchfile;
1625:C      3      highheap.a := userstack;
1626:C
1627:C      3      if not allresolved then escape(-119);
1628:C
1629:C      3      countcode;
1630:C
1631:C      3      highheap.a := userstack - totalglobal;
1632:C      3      if highheap.a < (a5 - 32768) then escape(117);
1633:C      3      startglobal := userstack - a5;
1634:C      3      userstack := highheap.a;
1635:C      3      zeromem(highheap.p, totalglobal);
1636:C
1637:C      3      loadtext(true);
1638:C
1639:C      3      movedefs(startreloc+totalreloc);
1640:C
1641:C      3      lowheap.a := startdefs + totaldefs;
1642:C      3      release(lowheap.p);
1643:C
1644:C      3      recover
1645:C      3      begin
1646:C      3          esc := escapecode; ior := ioreult;
1647:C      3          closefiles;
1648:C      3          releaseuser;
1649:C      3          ioreult := ior; escape(esc);
1650:C      3          end;
1651:C      2      end;
1652:C
1653:C      1      procedure initloader;
1654:C      2      begin
1655:C      2          sysdefs := addr(sysdeftable);
1656:C      2          {findroms;      {find and "load" rom modules }
1657:C      2          userstack := a5 + 32208;      (GBASE OF SYSTEM_P -- SEE *LINKMAP)
1658:C      2          markuser;
1659:C      2      end;
1660:C
1661:C      1      end
1662:C      2      $include 'LOR0ER' $;
1663:C
1664:C      1      import bootdammodule, sysglobals, asm, loader;
1665:C
1666:C      1      const splen = 20;
1667:C      1      type stringp = string[splen];
1668:C      1      const sysprefix = stringp['/WORKSTATIONS/SYSTEM'];
1669:C      1      var sysstart: string[11];
-12
1670:C
1671:C      1      procedure unitioinit;
1672:C      2      begin
1673:C      2          sysunit := 0;

```

```

1674:C      2      initbootdam;
1675:C      2      newwords(unitable, sizeof(unitentry) div 2);
1676:C      2      with unitable[0] do begin
1677:C      3          dam := bootdam;
1678:C      3          tm := boottm;
1679:C      3          sc := 0;
1680:C      3          ba := 0;
1681:C      3          du := 0;
1682:C      3          dv := 0;
1683:C      3          byteoffset := 0;
1684:C      3          devid := 0;
1685:C      3          uvld := ' ';
1686:C      3          dvrtmp := 0;      {must always initially be zero!}
1687:C      3          dvrtmp2 := -1;    {must always initially be -1!}
1688:C      3          letter := chr(0);
1689:C      3          offline := false;      {always initially assume online}
1690:C      3          uisinteractive := false;
1691:C      3          umediavalid := false;      {assume media invalid until verified}
1692:C      3          uuppercase := false;      {assume case is significant}
1693:C      3          uisfixed := false;      {assume unit not fixed}
1694:C      3          ureportchange := false;    {do not report media changes}
1695:C      3          pad := 0;      {not used yet}
1696:C      3          uisbkd := true;
1697:C      3          umaxbytes := 528 * fblksize;
1698:C      3      end (with)
1699:C
1700:C      3      end;
1701:C
1702:C      1      procedure initfkind;
1703:C      2      var fk: filekind;
-2
1704:C      2      begin
1705:C      2          mark(amtable);      {TO AVOID ADDRESSING ERRORS}
1706:C      2          new(efttable);
1707:C      2          for fk := untypedfile to lastfkind do efttable[fk] := 0;
1708:C      2          efttable[codefile] := -5582;
1709:C      2      end;
1710:C
1711:C      1      procedure go;
-1
1712:C      2      var done: boolean;
-6
1713:C      2      modptr: moddescptr;
1714:C      2      begin (G0)
1715:C      3      if entrypoint <> nil then
1716:C      4          try
1717:C      4              modptr := entrypoint;
1718:C      4              repeat
1719:C      5                  if modptr.startaddr > 0 then
1720:C      6                      userprogram(modptr.startaddr, userstack);
1721:C      5                  done := modptr.lastmodule;
1722:C      5                  modptr := modptr.link;
1723:C      5                  until done;
1724:C      4              recover escape(escapecode);
1725:C      2          {G0}
1726:C
1727:C      1      procedure add(var s: string; st: string80);
1728:C      2      var i: integer;
-88
1729:C      2      begin
1730:C      3      for i := 1 to strlen(st) do
1731:C      3          begin setstrlen(s, strlen(s)+1); s[strlen(s)] := st[i]; end;
1732:C      2      end;
1733:C

```



```

1734:D      1 procedure createsysname;
1735:D      2 type stringsysp = packed array[1..10] of char;
1736:D      2 const sysp      = stringsysp['SYSTEM_P'];
1737:D      2 var i,j: shortint;
1738:D      2 c: char;
1739:D      2
1740:C      2 begin
1741:C      2   sysstart := 'INITLI8';
1742:C      2   i := 0;
1743:C      2   repeat i := i + 1; until (i=10) or (sysname[i]<>sysp[i]);
1744:C      2   if i < 10 then
1745:C      3   begin
1746:C      3     setstrlen(sysstart, strmax(sysstart));
1747:C      3     if i < 8 then i := 3 else i := 7;
1748:C      3     j := 4;
1749:C      3     repeat
1750:C      4     i := i + 1;
1751:C      4     c := sysname[i];
1752:C      4     if c <> ' ' then begin j := j + 1; sysstart[j] := c; end;
1753:C      4     until (i=10) or (c=' ');
1754:C      3     setstrlen(sysstart, j);
1755:C      3   end;
1756:S      2 end;
1757:D      1 procedure trysrmboot;
1758:D      2 var s2: string2;
1759:D      2 il: string[40];
1760:C      2 begin
1761:C      2   s2 := bootnode;
1762:C      2   if strlen(s2) > 0 then
1763:C      3   begin
1764:C      3     nodestr := sysprefix; add(nodestr, s2); add(nodestr, '/');
1765:C      3     il := nodestr; add(il, sysstart);
1766:C      3     try loadq(il);
1767:C      3     recover if escapecode <> -10 then escape(escapecode);
1768:C      3     if entrypoint = nil then
1769:C      4     begin
1770:C      4       setstrlen(nodestr, splen+1); nodestr[splen+1] := '/';
1771:C      4       il := nodestr; add(il, sysstart);
1772:C      4       try loadq(il);
1773:C      4       recover if escapecode <> -10 then escape(escapecode);
1774:C      4     end;
1775:C      3   end;
1776:S      2 end;
1777:S      1 begin
1778:C      1   powerup;
1779:C      1   unitioinit;
1780:C      1   initfkinds;
1781:C      1   initloader;
1782:C      1
1783:S      1
1784:C      1   try
1785:C      2   createsysname;
1786:C      2   trysrmboot;
1787:C      2   if entrypoint = nil then
1788:C      3   begin
1789:C      3     nodestr := '';
1790:C      3     loadq(sysstart);
1791:C      3     end;
1792:C      2   cpymsg('');
1793:C      2   go;

```

```

(CLEAR SCREEN, PUT UP COPYRIGHT)
(EXECUTE ALL PROGRAMS IN SYSSTART)

```

```

1794:C      2   recover errmsg;
1795:C      1   while true do;
1796:C      1   end.
1797:S

```

```

(PRINT ERROR MESSAGE )

```

No errors. 2 warnings.

***** Nonstandard language features enabled *****

Description

IOLIB is the high-level, user-accessible I/O facilities.

Requirements

I/O library kernel (IODECLARATIONS, etc.)

© Copyright Hewlett-Packard Company. 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```
1:S      (*
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1984.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S     RESTRICTED RIGHTS LEGEND
11:S
12:S     Use, duplication, or disclosure by the Government
13:S     is subject to restrictions as set forth in
14:S     paragraph (b) (3) (8) of the Rights in Technical
15:S     Data and Computer Software clause in
16:S     DAR 7-104.9(a).
17:S
18:S     HEWLETT-PACKARD COMPANY
19:D     0 Fort Collins, Colorado      *)
20:S
21:S
22:D     0 $MODCAL ON$
23:D     0 $PARTIAL EVAL ON$
24:D     0 $STACKCHECK ON$
25:D     0 $RANGE OFF$
26:D     0 $DEBUG OFF$
27:D     0 $OVFLCHECK OFF$
```

```
28:D     0 $PAGES$
29:D     0 {*****}
30:D     0 {
31:D     0      RELEASED      VERSION      3.0
32:D     0 {
33:D     0 {*****}
34:D     0 {
35:D     0 {
36:D     0      IOLIB      IOLIB
37:D     0 {
38:D     0 {*****}
39:D     0 {
40:D     0 {
41:D     0 {
42:D     0 {      library      - IOLIB
43:D     0 {      name      - IOLIB
44:D     0 {      module(s) - general_1
45:D     0 {      - hpib_1
46:D     0 {      - general_2
47:D     0 {      - general_3
48:D     0 {      - general_4
49:D     0 {      - hpib_0
50:D     0 {      - hpib_2
51:D     0 {      - hpib_3
52:D     0 {      - serial_0
53:D     0 {      - serial_3
54:D     0 {
55:D     0 {      author      -
56:D     0 {      phone      -
57:D     0 {
58:D     0 {      date      - June 1, 1981
59:D     0 {      update     - June 4, 1984
60:D     0 {      release    - June 4, 1984
61:D     0 {
62:D     0 {      source      - IOLIB:IOLIB.TEXT
63:D     0 {      object     - IOLIB:IOLIB.CODE
64:D     0 {
65:D     0 {*****}
```



```
66:D 0 $PAGES$
67:D 0 *****
68:D 0 *
69:D 0 *
70:D 0 * This is the source code for an external procedures library
71:D 0 * to be used for general purpose interfacing on the HP 9826.
72:D 0 *
73:D 0 * The library consists of 3 primary sets of modules -
74:D 0 *
75:D 0 * 1. KERNEL modules
76:D 0 * 2. driver modules
77:D 0 * 3. 10LI8 modules
78:D 0 *
79:D 0 * The KERNEL modules consist of the following modules -
80:D 0 *
81:D 0 * 1. iodeclarations ( contains static r/w space )
82:D 0 * 2. iocomasm
83:D 0 * 3. general_0 ( initialization & low level
84:D 0 * routines like ioread/iowrite)
85:D 0 * The KERNEL modules also have an executable program segment
86:D 0 * that gets executed at the time it is loaded. This program
87:D 0 * initializes the static read/write memory. This program also
88:D 0 * allocates the temporary storage for any card that exists -
89:D 0 * independent of whether there is or is not a driver for it.
90:D 0 *
91:D 0 * The driver modules consist of the actual assembly or PASCAL
92:D 0 * routines that deal with a specific interface card. There is
93:D 0 * also an executable program segment for each driver module.
94:D 0 * This program searches the select code table in the static r/w
95:D 0 * initialized by the KERNEL general_0 module for all select codes
96:D 0 * that have the right interface card ( HPI8 drivers will search
97:D 0 * for the 98624 interface ). This program will then set up the
98:D 0 * driver tables to point to the correct drivers.
99:D 0 *
100:D 0 * The rest of the 10LI8 modules are high-level modules that are
101:D 0 * used by an end user in his/her application program.
102:D 0 *
103:D 0 * The KERNEL and some set of driver modules will exist in the
104:D 0 * SYSTEM.INITLI8 file as object code ( not EXPORT text ). The
105:D 0 * export text will reside on the SYSTEM.LIBRARY file. The rest
106:D 0 * of the library will reside on the SYSTEM.LIBRARY.
107:D 0 *
108:D 0 *****
```

```
109:D 0 $PAGES$
110:D 0 *****
111:D 0 *
112:D 0 *
113:D 0 * BUG FIX HISTORY - after release
114:D 0 *
115:D 0 *
116:D 0 * BUG # 8Y / ON LOCATION DESCRIPTION
117:D 0 * -----
118:D 0 *
119:D 0 * 1250 ----- HPI8_3 request service allows
120:D 0 * 01/08/1982 request_service active ctl to request
121:D 0 * service.
122:D 0 *
123:D 0 * 1251 ----- HPI8_2 local(7) with sc 7 as
124:D 0 * 01/08/1982 local sys ctl / not act ctl
125:D 0 * 01/26/1982 gives error.
126:D 0 *
127:D 0 * 1252 ----- HPI8_2 remote(7) with sc 7 as
128:D 0 * 01/08/1982 remote act ctl / not sys ctl
129:D 0 * doesn't give error.
130:D 0 *
131:D 0 * 1258 ----- HPI8_2 pass_control pass control sends the
132:D 0 * 01/08/1982 wrong sequence to pass
133:D 0 * control to itself.
134:D 0 *
135:D 0 * 1269 ----- SERIAL_3 bad check for a 98626
136:D 0 * 01/08/1982 set_stop_bits card.
137:D 0 *
138:D 0 * 1270 ----- SERIAL_3 make procedures for
139:D 0 * 01/08/1982 set_baud_rate data comm consistent
140:D 0 * set_stop_bits for buffered control.
141:D 0 * set_parity
142:D 0 * set_char_length
143:D 0 *
144:D 0 * 1281 ----- GENERAL_3 Wrong message for error
145:D 0 * 01/08/1982 ioerror_message io_not_dvc.
146:D 0 *
147:D 0 * 0082 ----- GENERAL_3 Addition of a link for
148:D 0 * 07/23/1982 ioerror_message the error messages.
149:D 0 * See also IODECLARATIONS.
150:D 0 *
151:D 0 * 0083 ----- GENERAL_4 Addition of buffer_busy
152:D 0 * 07/23/1982 buffer_busy and isc_busy routines.
153:D 0 * isc_busy
154:D 0 *
155:D 0 * 0355 ----- SERIAL_3 Set parity of one and
156:D 0 * 08/20/1982 set_parity zero parity is backwards
157:D 0 * for the 98628 card.
158:D 0 *
159:D 0 * 0359 ----- SERIAL_3 Changes for addition
160:D 0 * 08/26/1982 set_parity of 98626 drivers.
161:D 0 * set_char_length
162:D 0 * set_stop_bits
163:D 0 *
164:D 0 * 0364 ----- GENERAL_3 Addition of SRM driver
165:D 0 * 08/23/1982 ioerror_message error codes. See also
166:D 0 * IODECLARATIONS.
167:D 0 *
168:D 0 * 557 ----- GENERAL_3 Mistyped. ( typo )
```



```

169:D 0 (* 10/01/1982 set_parity *)
170:D 0 (* *)
171:D 0 (* jsjs ----- HP1B_2 BUG FIX error in Local *)
172:D 0 (* 03/09/1983 local procedure for isc param *)
173:D 0 (* *) and not sys controller. *)
174:D 0 (* *)
175:D 0 (* tttt ----- HP1B_1 Use timer on CPU board *)
176:D 0 (* 08/03/1983 if available for timeout *)
177:D 0 (* *) checking *)
178:D 0 (* *)
179:D 0 (* ----- serial modules add code for 98644 *)
180:D 0 (* 5/15/84 *)
181:D 0 (* 6/4/84 *)
182:D 0 (*****

```

```

183:D 0 $PAGE$
184:D 0 (*****
185:D 0 (*
186:D 0 (*
187:D 0 (* REFERENCES :
188:D 0 (*
189:D 0 (*
190:D 0 (* 1. 9826 I/O Designers Guide ( ----- )
191:D 0 (*
192:D 0 (* 2. 68000 Manual ( Motorola )
193:D 0 (*
194:D 0 (* 3. Pascal alpha site ERS ( ----- )
195:D 0 (*
196:D 0 (* 4. Pascal I/O Library ERS ( ----- )
197:D 0 (*
198:D 0 (* 5. 9826 HPL E10 & IOD listings ( ----- )
199:D 0 (*
200:D 0 (* 6. 9826 HPL Misc. I/O Doc. ( ----- )
201:D 0 (*
202:D 0 (* 7. 9826 card documentation ( ----- )
203:D 0 (*
204:D 0 (* 8. Pascal I/O Library IRS ( ----- )
205:D 0 (*
206:D 0 (*
207:D 0 (*
208:D 0 (*****

```



```

209:D 0 $PAGES
210:D 0 {*****}
211:D 0 {*****}
212:D 0 {*****}
213:D 0 {*****}
214:D 0 {*****}
215:D 0 {*****}
216:D 0 {*****}
217:D 0 {*****}
218:D 0 {*****}
219:D 0 {*****}
220:D 0 MODULE general_1 ;
221:D 0
222:D 0 {
223:D 0   date   07/15/81
224:D 0   update 11/20/81
225:D 0 }
226:D 0   purpose This module contains the LEVEL 1 GENERAL GROUP procedures.
227:D 0
228:D 0
229:D 0
230:D 1 IMPORT $SEARCH 'IOLIB:KERNEL.CODE', 'IOLIB:COMASM'$
231:D 1   iodeclarations ;
232:D 1
233:D 1 EXPORT
234:D 1
235:D 1   PROCEDURE ioinitialize;
236:D 1   PROCEDURE iouninitialize;
237:D 1   PROCEDURE ioreset ( select_code : type_isc;
238:D 1   PROCEDURE readchar ( select_code : type_isc ;
239:D 2   VAR value : CHAR );
240:D 1   PROCEDURE writechar ( select_code : type_isc ;
241:D 2   value : CHAR );
242:D 1   PROCEDURE readword ( select_code : type_isc ;
243:D 2   VAR num : INTEGER );
244:D 1   PROCEDURE writeword ( select_code : type_isc ;
245:D 2   value : INTEGER );
246:D 1   PROCEDURE set_timeout ( select_code : type_isc ;
247:D 2   time : REAL );
248:D 1
249:D 1
250:D 1 IMPLEMENT
251:D 1
252:D 1   IMPORT general_0;
253:D 1
254:D 1
255:D 1   PROCEDURE ioinitialize;
256:D 2 BEGIN
257:D 2   io_system_reset;
258:D 2
259:D 2   END; { of ioinitialize }
260:D 2
261:D 2
262:D 2
263:D 2
264:D 2
265:D 1   PROCEDURE iouninitialize;
266:D 2 BEGIN
267:D 2   io_system_reset;
268:D 2

```

```

269:D 2   END; { of iouninitialize }
270:D 2
271:D 2
272:D 2
273:D 2
274:D 1   PROCEDURE ioreset ( select_code : type_isc;
275:D 2 BEGIN
276:D 2   WITH isc_table[select_code] DO
277:D 3   CALL(io_drv_ptr^.iod_init,
278:D 3   io_tmp_ptr);
279:D 2   END; { of ioreset }
280:D 2
281:D 2
282:D 1   PROCEDURE readchar ( select_code : type_isc ;
283:D 2   VAR value : CHAR );
284:D 2 BEGIN
285:D 2   WITH isc_table[select_code] DO
286:D 3   CALL(io_drv_ptr^.iod_rdb,
287:D 3   io_tmp_ptr,
288:D 3   value);
289:D 2   END; { of readchar }
290:D 2
291:D 2
292:D 2
293:D 1   PROCEDURE writechar ( select_code : type_isc ;
294:D 2   value : CHAR );
295:D 2 BEGIN
296:D 2   WITH isc_table[select_code] DO
297:D 3   CALL(io_drv_ptr^.iod_wtb,
298:D 3   io_tmp_ptr,
299:D 3   value);
300:D 2   END; { of writechar }
301:D 2
302:D 2
303:D 2
304:D 1   PROCEDURE readword ( select_code : type_isc ;
305:D 2   VAR num : INTEGER );
306:D -2 2   VAR my_num : io_word;
307:D 2 BEGIN
308:D 2   WITH isc_table[select_code] DO
309:D 3   CALL(io_drv_ptr^.iod_rdw,
310:D 3   io_tmp_ptr,
311:D 3   my_num);
312:D 2   num:=my_num;
313:D 2   END; { of readword }
314:D 2
315:D 2
316:D 1   PROCEDURE writeword ( select_code : type_isc ;
317:D 2   value : INTEGER );
318:D -2 2   VAR my_value : io_word;
319:D 2 BEGIN
320:D 2   my_value:=value;
321:D 2   WITH isc_table[select_code] DO
322:D 3   CALL(io_drv_ptr^.iod_wtw,
323:D 3   io_tmp_ptr,
324:D 3   my_value);
325:D 2   END; { of writeword }
326:D 2
327:D 2
328:D 2

```



```

329:S
330:D  -8 1  PROCEDURE set_timeout ( select_code : type_isc ;
331:D      2      time : REAL ; { in seconds } );
332:C  2  BEGIN
333:C      2  IF time>8191 { 4 byte timeout - 1 byte left for shifts }
334:C      3  THEN BEGIN
335:C      3  { error }
336:C      3  io_escape(ioe_bad_tmo,select_code);
337:C      3  END; { of IF }
338:C      2  IF (time>0) AND (time<0.001)
339:C      3  THEN BEGIN
340:C      3  { error }
341:C      3  io_escape(ioe_bad_tmo,select_code);
342:C      3  END; { of IF }
343:S
344:C      2  WITH isc_table[select_code] DO BEGIN
345:C      3  { the table entry used by drivers is in milliseconds }
346:C      3  user_time:=ROUND(time*1000);
347:C      3
348:S      3  IF io_tmp_ptr <> NIL THEN io_tmp_ptr^.timeout := user_time;
349:C      3  END; { of WITH DO BEGIN }
350:S
351:C      2  END; { of set_timeout }
352:S
353:S
354:S
355:S
356:C      1  END, { of general_1 }

```

```

357:D  1  $PAGE$
358:D  1  {*****}
359:D  1  {*}
360:D  1  {*}
361:D  1  {*      HPIB GROUP LEVEL 1}
362:D  1  {*}
363:D  1  {*}
364:D  1  {*****}
365:D  1  {*}
366:D  1  {*}
367:D  1  {*      This level is included in the}
368:D  1  {*      general group because HP-IB}
369:D  1  {*      addressing is necessary for}
370:D  1  {*      general purpose device speci-}
371:D  1  {*      fication.}
372:D  1  {*}
373:D  1  {*****}
374:S
375:S
376:S
377:D  1  MODULE hpib_1 ;
378:S
379:S      (
380:S          date 07/16/81
381:S          update 08/03/83
382:S
383:S          purpose This module contains the LEVEL 1 HPIB GROUP procedures.
384:S
385:D      1  )
386:S
387:S
388:D  1  IMPORT iodeclarations ;
389:S
390:D  1  EXPORT
391:S
392:S
393:D  1  PROCEDURE send_command( select_code : type_isc ;
394:D      2      command : CHAR );
395:D  1  FUNCTION my_address ( select_code : type_isc)
396:D      2      : type_hpib_addr ;
397:D  1  FUNCTION active_controller
398:D      2      ( select_code : type_isc)
399:D      2      : BOOLEAN;
400:D  1  FUNCTION system_controller
401:D      2      ( select_code : type_isc)
402:D      2      : BOOLEAN;
403:D  1  FUNCTION addr_to_talk( device : type_device)
404:D      2      : type_isc;
405:D  1  FUNCTION addr_to_listen
406:D      2      ( device : type_device)
407:D      2      : type_isc;
408:D  1  FUNCTION set_to_talk ( device : type_device)
409:D      2      : type_isc;
410:D  1  FUNCTION set_to_listen
411:D      2      ( device : type_device)
412:D      2      : type_isc;
413:D  1  FUNCTION end_set ( select_code : type_isc )
414:D      2      : BOOLEAN;
415:S
416:S

```



```

417:D      1 IMPLEMENT
418:S
419:D      1  IMPORT  iocomasm ,
420:D      1          general_0 ;
421:S
422:D      1  TYPE    timeoutrec = record                { tttt JS 8/3/83 }
423:D      1          counter: integer;                  { tttt JS 8/3/83 }
424:D      1          firsttime: boolean;                { tttt JS 8/3/83 }
425:D      1          end;                                { tttt JS 8/3/83 }
426:S
427:S
428:D      1  FUNCTION timerexists: boolean; external;    { tttt JS 8/3/83 }
429:S
430:D      1  FUNCTION timed_out(var rec: timeoutrec): boolean; external; { tttt JS 8/3/83 }
431:S
432:D      1  PROCEDURE send_command( select_code : type_isc ;
433:D      2          command : CHAR );
434:C      2  BEGIN
435:C      3      WITH isc_table[select_code] DO
436:C      3          CALL ( io_drv_ptr^.iod_send,
437:C      3              io_tmp_ptr,
438:C      3              command );
439:C      2  END; { of send_command }
440:S
441:S
442:S
443:D      1  FUNCTION my_address ( select_code : type_isc)
444:D      2          : type_hpib_addr ;
445:C      2  BEGIN
446:C      3      IF isc_table[select_code].io_tmp_ptr <> NIL
447:C      3      THEN BEGIN
448:C      4          WITH isc_table[select_code].io_tmp_ptr^ DO
449:C      4              IF addressed <> -1
450:C      4              THEN BEGIN
451:C      5                  my_address:=addressed;
452:C      5              END
453:C      5              ELSE BEGIN
454:C      5                  { error }
455:C      5                  io_escape(ioe_not_hpib,select_code);
456:C      5              END; { of IF addressed }
457:C      3      END
458:C      3      ELSE BEGIN
459:C      3          { error }
460:C      3          io_escape(ioe_not_hpib,select_code);
461:C      3      END; { of IF io_tmp_ptr }
462:C      2  END; { of my_address }
463:S
464:S
465:S
466:D      1  FUNCTION active_controller
467:D      2          ( select_code : type_isc)
468:D      2          : BOOLEAN;
469:C      2  BEGIN
470:C      3      IF isc_table[select_code].card_type=hpib_card
471:C      3      THEN BEGIN
472:C      3          active_controller:=bit_set(iostatus(select_code,3),6);
473:C      3      END
474:C      3      ELSE BEGIN
475:C      3          active_controller := TRUE;
476:C      3      END; { of IF }

```

```

477:C      2  END; { of active_controller }
478:S
479:S
480:S
481:D      1  FUNCTION system_controller
482:D      2          ( select_code : type_isc)
483:D      2          : BOOLEAN;
484:C      2  BEGIN
485:C      3      IF isc_table[select_code].card_type=hpib_card
486:C      3      THEN BEGIN
487:C      3          system_controller:=bit_set(iostatus(select_code,3),7);
488:C      3      END
489:C      3      ELSE BEGIN
490:C      3          system_controller := TRUE;
491:C      3      END; { of IF }
492:C      2  END; { of system_controller }
493:S
494:S
495:S
496:D      1  FUNCTION end_set      ( select_code : type_isc )
497:D      2          : BOOLEAN ;
498:D      -1 2  VAR mybool : BOOLEAN;
499:C      2  BEGIN
500:C      3      WITH isc_table[select_code] DO
501:C      3          CALL ( io_drv_ptr^.iod_end,
502:C      3              io_tmp_ptr,
503:C      3              mybool );
504:C      2      end_set := mybool;
505:C      2  END; { of send_command }
506:S
507:S

```



```

508:D      1  $PAGE$
509:S      1
510:D      2  FUNCTION addr_to_talk( device      : type_device)
511:D      2
512:D      2  VAR io_isc : type_isc;
513:D      2  timer : INTEGER;
514:D      2  hpbrec: timeoutrec;
515:S      2  {tttt JS 8/3/83}
516:C      2  BEGIN
517:S      2
518:C      2  IF device>iomaxisc
519:C      2  THEN BEGIN
520:C      3  io_isc:=device DIV 100;
521:S      3
522:C      3  WITH isc_table[io_isc] DO BEGIN
523:S      3
524:C      4  IF io_tmp_ptr <> NIL
525:C      5  THEN BEGIN
526:S      5
527:C      5  { set up user timeout - in case system drivers changed it }
528:C      5  io_tmp_ptr^.timeout:=user_time;
529:S      5
530:C      5  IF io_tmp_ptr^.addressed <> -1
531:C      6  THEN BEGIN
532:C      6  IF ( card_type <> hpb_card ) AND
533:C      6  ( device MOD 100 > 31 )
534:C      7  THEN io_escape(ioe_misc,io_isc);
535:C      6  send_command(io_isc,CHR(talk_constant+(device MOD 100)));
536:C      6  send_command(io_isc,'?');
537:C      6  send_command(io_isc,CHR(my_address(io_isc)+listen_constant));
538:C      6  END
539:C      6  ELSE BEGIN
540:C      6  { error }
541:C      6  io_escape(ioe_not_hpb,io_isc);
542:C      6  END; { of IF }
543:C      5  END
544:C      5  ELSE BEGIN
545:C      5  END; { of IF }
546:C      4  END; { of WITH DO BEGIN }
547:C      3  END
548:C      3  ELSE BEGIN
549:C      3  io_isc:=device;
550:C      3
551:C      3  WITH isc_table[io_isc] DO BEGIN
552:S      3
553:C      4  { set up user timeout - in case system drivers changed it }
554:C      4  IF io_tmp_ptr <> NIL THEN io_tmp_ptr^.timeout:=user_time;
555:S      4
556:C      5  IF card_type=hpb_card THEN
557:C      5  BEGIN
558:C      5  IF NOT active_controller(io_isc)
559:C      6  THEN BEGIN
560:C      6  { if non controller wait until listener }
561:C      6  IF user_time = 0
562:C      7  THEN BEGIN
563:C      7  REPEAT
564:C      8  { wait forever }
565:C      8  UNTIL bit_set(iostatus(io_isc,6),10);
566:C      7  END
567:C      5  ELSE BEGIN

```

```

568:C      7  { wait for timeout value }
569:C      7  IF timereexists THEN BEGIN
570:C      8  hpbrec.firsttime:=true;
571:C      8  hpbrec.counter:=user_time;
572:C      8  REPEAT
573:C      9  UNTIL timed_out(hpbrec) OR
574:C      9  bit_set(iostatus(io_isc,6),10);
575:C      8  END
576:C      8  ELSE BEGIN
577:C      8  timer:=user_time*3;
578:C      8  REPEAT
579:C      9  timer:=timer-1;
580:C      9  UNTIL ( timer = 0 ) OR
581:C      9  ( bit_set(iostatus(io_isc,6),10) );
582:C      8  END;
583:C      7  IF NOT bit_set(iostatus(io_isc,6),10)
584:C      8  THEN io_escape(ioe_timeout,io_isc);
585:C      7  END; { of IF user_time=0 }
586:C      6  END; { of IF }
587:C      5  END; { of IF card_type = hpb_card }
588:C      4  END; { of WITH DO BEGIN }
589:C      3  END; { of IF }
590:S      3
591:C      2  addr_to_talk:=io_isc; { return select code }
592:S      2
593:C      2  END; { of addr_to_talk }
594:S      2

```



```

595:D      1  $PAGE$
596:S
597:D      1  FUNCTION addr_to_listen
598:D      2      ( device      : type_device)
599:D      2      : type_is;
600:D      -2  VAR io_isc : type_is;
601:D      -6  timer : INTEGER;
602:D      -12 2      hpbrec: timeoutrec;          {tttt JS 8/3/83}
603:S
604:C      2  BEGIN
605:S
606:C      2      IF device>iomaxisc
607:C      3      THEN BEGIN
608:C      3          io_isc:=device DIV 100;
609:S
610:C      3          WITH isc_table[io_isc] DO BEGIN
611:S
612:C      4              IF io_tmp_ptr <> NIL
613:C      5              THEN BEGIN
614:S
615:C      5                  { set up user timeout - in case system drivers changed it }
616:C      5                  io_tmp_ptr^.timeout:=user_time;
617:S
618:C      5                  IF io_tmp_ptr^.addressed <> -1
619:C      6                  THEN BEGIN
620:C      6                      IF ( card_type <> hpib_card ) AND
621:C      7                      ( device MOD 100 > 31 )
622:C      7                      THEN io_escape(io_misc,io_isc);
623:C      6                      send_command(io_isc,CHR(my_address(io_isc)+talk_constant));
624:C      6                      send_command(io_isc,'?');
625:C      6                      send_command(io_isc,CHR(listen_constant+(device MOD 100)));
626:C      6                      END
627:C      6                      ELSE BEGIN
628:C      6                          { error }
629:C      6                          io_escape(io_not_hpib,io_isc);
630:C      6                      END; { of IF }
631:C      5                  END
632:C      5                  ELSE BEGIN
633:C      5                      END; { of IF }
634:C      4                  END; { of WITH DO BEGIN }
635:C      3              END
636:C      3              ELSE BEGIN
637:C      3                  io_isc:=device;
638:C      3              WITH isc_table[io_isc] DO BEGIN
639:C      4                  { set up user timeout - in case system drivers changed it }
640:C      4                  IF io_tmp_ptr <> NIL THEN io_tmp_ptr^.timeout:=user_time;
641:S
642:C      4                  IF card_type=hpib_card THEN
643:C      5                  BEGIN
644:C      5                      IF NOT active_controller(io_isc)
645:C      6                      THEN BEGIN
646:C      6                          { if non controller wait until talker }
647:C      6                          IF user_time = 0
648:C      7                          THEN BEGIN
649:C      7                              REPEAT
650:C      7                                  { wait forever }
651:C      8                                  UNTIL bit_set(iostatus(io_isc,6),9);
652:C      8                              END
653:C      8                              REPEAT
654:C      7                              UNTIL bit_set(iostatus(io_isc,6),9);
655:C      7                              END
656:C      7                              { wait for timeout value }
657:C      7                              IF timerexists THEN BEGIN          {tttt JS 8/3/83}
658:C      8                                  hpbrec.firsttime:=true;          {tttt JS 8/3/83}
659:C      8                                  hpbrec.counter:=user_time;      {tttt JS 8/3/83}
660:C      8                                  REPEAT                          {tttt JS 8/3/83}
661:C      9                                  UNTIL timed_out(hpbrec) OR      {tttt JS 8/3/83}
662:C      9                                  bit_set(iostatus(io_isc,6), 9);  {tttt JS 8/3/83}
663:C      8                                  END                          {tttt JS 8/3/83}
664:C      8                                  ELSE BEGIN                          {tttt JS 8/3/83}
665:C      8                                      timer:=user_time*3;
666:C      8                                      REPEAT
667:C      9                                      timer:=timer-1;
668:C      9                                      UNTIL ( timer = 0 ) OR
669:C      9                                      ( bit_set(iostatus(io_isc,6),9) );
670:C      8                                      END; {tttt JS 8/3/83}
671:C      7                                      IF NOT bit_set(iostatus(io_isc,6),9)
672:C      7                                      THEN io_escape(io_timeout,io_isc);
673:C      7                                      END; { of IF user_time=0 }
674:C      6                                  END; { of IF }
675:C      5                                  END; { of IF card_type = hpib_card }
676:C      4                                  END; { of WITH DO BEGIN }
677:C      3                                  END; { of IF }
678:S
679:C      2      addr_to_listen:=io_isc;
680:S
681:C      2      END; { of addr_to_listen }
682:S

```

```

655:C      7      ELSE BEGIN
656:C      7          { wait for timeout value }
657:C      7          IF timerexists THEN BEGIN          {tttt JS 8/3/83}
658:C      8              hpbrec.firsttime:=true;          {tttt JS 8/3/83}
659:C      8              hpbrec.counter:=user_time;      {tttt JS 8/3/83}
660:C      8              REPEAT                          {tttt JS 8/3/83}
661:C      9              UNTIL timed_out(hpbrec) OR      {tttt JS 8/3/83}
662:C      9              bit_set(iostatus(io_isc,6), 9);  {tttt JS 8/3/83}
663:C      8              END                          {tttt JS 8/3/83}
664:C      8              ELSE BEGIN                          {tttt JS 8/3/83}
665:C      8                  timer:=user_time*3;
666:C      8                  REPEAT
667:C      9                  timer:=timer-1;
668:C      9                  UNTIL ( timer = 0 ) OR
669:C      9                  ( bit_set(iostatus(io_isc,6),9) );
670:C      8                  END; {tttt JS 8/3/83}
671:C      7                  IF NOT bit_set(iostatus(io_isc,6),9)
672:C      7                  THEN io_escape(io_timeout,io_isc);
673:C      7                  END; { of IF user_time=0 }
674:C      6                  END; { of IF }
675:C      5                  END; { of IF card_type = hpib_card }
676:C      4                  END; { of WITH DO BEGIN }
677:C      3                  END; { of IF }
678:S
679:C      2      addr_to_listen:=io_isc;
680:S
681:C      2      END; { of addr_to_listen }
682:S

```



```

683:D      1  $PAGE$
684:S
685:S
686:S      ( set to talk exists because of HPIB_2/HPIB_3 -
687:S      those routines are intended to be the controller
688:S      ( active ) and should not wait for the card to be
689:S      addressed as talker.  addr_to_talk is used by
690:S      data transfer routines.  set_to_talk is used by
691:D      1  bus control routines.  )
692:S
693:S
694:S
695:S
696:D      1  FUNCTION set_to_talk ( device      : type_device)
697:D      2      ( device      : type_isc;
698:D      -2 2  VAR io_isc : type_isc;
699:C      2  BEGIN
700:S
701:C      2      IF device>iomaxisc
702:C      3      THEN BEGIN
703:C      3          io_isc:=addr_to_talk(device);
704:C      3      END
705:C      3      ELSE BEGIN
706:C      3          io_isc:=device;
707:C      3
708:C      3      WITH isc_table[io_isc] DO BEGIN
709:S
710:C      4          { set up user timeout - in case system drivers changed it }
711:C      4          IF io_tmp_ptr <> NIL THEN io_tmp_ptr^.timeout:=user_time;
712:S
713:C      4          IF card_type=hpib_card THEN
714:C      5              BEGIN
715:C      5                  IF NOT active_controller(io_isc)
716:C      6                      THEN BEGIN
717:S
718:C      6                      io_escape(ioe_not_act,io_isc);
719:S
720:C      6                      END; { of IF }
721:C      5                  END; { of IF card_type = hpib_card }
722:C      4                  END; { of WITH DO BEGIN }
723:C      3                  END; { of IF }
724:S
725:C      2          set_to_talk:=io_isc; { return select code }
726:S
727:C      2      END; { of set_to_talk }
728:S

```

```

729:D      1  $PAGE$
730:S
731:S
732:S      ( set to listen exists because of HPIB_2/HPIB_3 -
733:S      those routines are intended to be the controller
734:S      ( active ) and should not wait for the card to be
735:S      addressed as listener.  addr_to_listen is used by
736:S      data transfer routines.  set_to_listen is used by
737:D      1  bus control routines.  )
738:S
739:S
740:S
741:S
742:S
743:D      1  FUNCTION set_to_listen
744:D      2      ( device      : type_device)
745:D      2      ( device      : type_isc;
746:D      -2 2  VAR io_isc : type_isc;
747:D      -6 2  timer : INTEGER;
748:C      2  BEGIN
749:S
750:C      2      IF device>iomaxisc
751:C      3      THEN BEGIN
752:C      3          io_isc:=addr_to_listen(device);
753:S
754:C      3      END
755:C      3      ELSE BEGIN
756:C      3          io_isc:=device;
757:C      3
758:C      3      WITH isc_table[io_isc] DO BEGIN
759:S
760:C      4          { set up user timeout - in case system drivers changed it }
761:C      4          IF io_tmp_ptr <> NIL THEN io_tmp_ptr^.timeout:=user_time;
762:S
763:C      4          IF card_type=hpib_card THEN
764:C      5              BEGIN
765:C      5                  IF NOT active_controller(io_isc)
766:C      6                      THEN BEGIN
767:S
768:C      6                      io_escape(ioe_not_act,io_isc);
769:S
770:C      6                      END; { of IF }
771:C      5                  END; { of IF card_type = hpib_card }
772:C      4                  END; { of WITH DO BEGIN }
773:C      3                  END; { of IF }
774:S
775:C      2          set_to_listen:=io_isc;
776:S
777:C      2      END; { of set_to_listen }
778:S
779:S
780:S
781:C      1  END; { of hpib_1 }

```



```

782:D      1 $PAGES
783:D      1 MODULE general_2 ;
784:S
785:S      {
786:S          date 07/15/81
787:S          update 11/30/81
788:S
789:S          purpose This module contains the LEVEL 2 GENERAL GROUP procedures.
790:S
791:D      1      }
792:S
793:D      1 IMPORT   iodeclarations;
794:S
795:D      1 EXPORT
796:D      1
797:D      1   PROCEDURE readnumber ( device : type_device ;
798:D      1                               VAR num: REAL ) ;
799:D      1   PROCEDURE writenumber( device : type_device ;
800:D      2                               value : REAL ) ;
801:D      1   PROCEDURE readstring
802:D      2       ( device : type_device ;
803:D      2         VAR str: STRING ) ;
804:D      1   PROCEDURE readstring_until
805:D      2       ( term : CHAR ;
806:D      2         device : type_device ;
807:D      2         VAR str: STRING ) ;
808:D      1   PROCEDURE writestring( device : type_device ;
809:D      2                               str : io_STRING ) ;
810:D      1   PROCEDURE readnumberln
811:D      2       ( device : type_device ;
812:D      2         VAR num: REAL ) ;
813:D      1   PROCEDURE writenumberln
814:D      2       ( device : type_device ;
815:D      2         value : REAL ) ;
816:D      1   PROCEDURE writestringln
817:D      2       ( device : type_device ;
818:D      2         str : io_STRING ) ;
819:D      1   PROCEDURE readuntil ( term : CHAR ;
820:D      2         device : type_device ) ;
821:D      1   PROCEDURE skipfor ( count : INTEGER ;
822:D      2         device : type_device ) ;
823:S
824:S
825:D      1 IMPLEMENT
826:D      1 IMPORT   sysglobals,
827:D      1         hpib_1,
828:D      1         general_1 ;
829:S
830:S
831:D      1
832:D      1   PROCEDURE readnumber ( device : type_device ;
833:D      2                               VAR num: REAL ) ;
834:D      2   VAR io_work_str : STRING[255];
835:D      2   i : INTEGER;
836:D      2   p2 : INTEGER;
837:D      2   io_isc : type_isc;
838:D      2   numbuilt : BOOLEAN;
839:S
840:D      2   EUNCTION numeric ( character : CHAR ) : BOOLEAN;
841:C      3   BEGIN

```

```

842:C      3   CASE character OF
843:C      4       '0'..'9' :
844:C      4       'E'..'e' : numeric:=TRUE
845:C      4
846:C      4       OTHERWISE numeric:=FALSE
847:C      4   END; { of CASE }
848:C      3   END; { of numeric }
849:C
850:C      2 BEGIN
851:C      2   { use TRY RECOVER to build a number until I find one }
852:S
853:C      2   io_isc:=addr_to_talk(device);
854:S
855:C      2   numbuilt := FALSE;
856:S
857:C      2   WITH isc_table[io_isc].io_drv_ptr^ DO BEGIN
858:C      3       isc_table[io_isc]
859:S
860:C      3   REPEAT
861:C      4
862:C      4       SETSTRLEN(io_work_str,255);
863:C      4       i:=1;
864:S
865:C      4       { skip over non-numeric characters }
866:C      4       REPEAT
867:C      5           CALL ( io_rdb , io_tmp_ptr , io_work_str[i] );
868:C      5           WHILE io_work_str[i]="" DO
869:C      6               CALL ( io_rdb , io_tmp_ptr , io_work_str[i] );
870:C      5           UNTIL numeric(io_work_str[i] );
871:S
872:C      4       { read in numeric characters }
873:C      4       REPEAT
874:C      5           i:=i+1;
875:C      5           CALL ( io_rdb , io_tmp_ptr , io_work_str[i] );
876:C      5           WHILE io_work_str[i]="" DO
877:C      6               CALL ( io_rdb , io_tmp_ptr , io_work_str[i] );
878:C      5           UNTIL ( (NOT (numeric(io_work_str[i]))) OR (
879:C      5               i>=255) ) );
880:S
881:C      4       SETSTRLEN(io_work_str,i);
882:C      4       io_work_char:=io_work_str[i];
883:C
884:S      4   TRY
885:C      5       STRREAD(io_work_str,1,p2,num);
886:C
887:C      5       numbuilt := TRUE;
888:C
889:C      5   RECOVER BEGIN
890:C      5       IF ( ESCAPECODE=-10 ) AND
891:C      6           ( ( IORESULT = ORD(ISOFORMAT) ) OR
892:C      6             ( IORESULT = ORD(ISTROVFL) ) ) )
893:C      6           THEN BEGIN
894:C      6               { this is the streadd errors - try again }
895:C      6               END;
896:C      6           ELSE BEGIN
897:C      6               { this means something else happened }
898:C      6               ESCAPE(ESCAPECODE);
899:C      6           END; { of IE my error }
900:C
901:C      6

```



```

902:C      5      END; { of RECDVER }
903:S
904:C      4      UNTIL numbult;
905:S
906:C
907:C      3      ENO; { of WITH DD BEGIN }
908:S
909:C      2      END; { of readnumber }
910:S
911:S
912:S
913:D      1      PROCEDURE writenumber (device : type_device ;
914:D      1      value : REAL);
915:D      1      VAR i      : INTEGER;
916:D      1      p2      : INTEGER;
917:D      1      io_isc   : type_isc;
918:D      1      io_work_str : STRING[255];
919:D      1      BEGIN
920:D      1      io_isc:=addr_to_listen(device);
921:D      1
922:D      1      WITH isc_table[io_isc].io_drv_ptr^ DO BEGIN
923:D      1      isc_table[io_isc]
924:D      1
925:D      1      STRWRITE(io_work_str,1,p2,value);
926:D      1      FOR i:=1 TO p2-1 DO
927:D      1      CALL ( iod_wtb , io_tmp_ptr , io_work_str[i]);
928:D      1
929:D      1      END; { of WITH DD }
930:D      1
931:D      1      END; { of writenumber }
932:S
933:S
934:S
935:D      1      PROCEDURE readstring
936:D      1      ( device : type_device ;
937:D      1      VAR str: STRING );
938:D      1      VAR i      : INTEGER;
939:D      1      io_isc   : type_isc;
940:D      1      BEGIN
941:D      1      io_isc:=addr_to_talk(device);
942:D      1
943:D      1      WITH isc_table[io_isc].io_drv_ptr^
944:D      1      isc_table[io_isc] DO BEGIN
945:D      1
946:D      1      SETSTLEN(str,STRMAX(str));      { so I can do assign to empty string }
947:D      1      i:=0;
948:D      1      REPEAT
949:D      1      i:=i+1;
950:D      1      CALL ( iod_rdb , io_tmp_ptr , str[i]);
951:D      1      UNTIL ( (i>=STRMAX(str)) OR
952:D      1      ( str[i] = io_line_feed ));
953:D      1      IF i<>0 THEN IF str[i]=io_carriage_rtn THEN i:=i-1;
954:D      1      SETSTLEN(str,i);
955:D      1
956:D      1      END; { of WITH DO BEGIN }
957:D      1
958:D      1      END; { of readstring }
959:S
960:S
961:S

```

```

962:S
963:S
964:D      1      PROCEDURE readstring_until
965:D      1      ( term : CHAR ;
966:D      1      device : type_device ;
967:D      1      VAR str: STRING );
968:D      1      VAR i      : INTEGER;
969:D      1      io_isc   : type_isc;
970:D      1      io_work_char: CHAR;
971:D      1      BEGIN
972:D      1      io_isc:=addr_to_talk(device);
973:D      1
974:D      1      WITH isc_table[io_isc].io_drv_ptr^
975:D      1      isc_table[io_isc] DO BEGIN
976:D      1
977:D      1      SETSTLEN(str,STRMAX(str));
978:D      1      i:=0;
979:D      1      REPEAT
980:D      1      i:=i+1;
981:D      1      CALL ( iod_rdb , io_tmp_ptr , str[i]);
982:D      1      UNTIL ( (i>=STRMAX(str)) OR
983:D      1      ( str[i]=term ));
984:D      1      SETSTLEN(str,i);
985:D      1
986:D      1      END; { of WITH DO BEGIN }
987:D      1
988:D      1      END; { of readstring_until }
989:S
990:S
991:S
992:D      1      PROCEDURE writestring( device : type_device ;
993:D      1      str : io_STRING );
994:D      1      VAR i      : INTEGER;
995:D      1      io_isc: type_isc;
996:D      1      BEGIN
997:D      1      io_isc:=addr_to_listen(device);
998:D      1
999:D      1      WITH isc_table[io_isc].io_drv_ptr^
1000:D      1      isc_table[io_isc] DO BEGIN
1001:D      1
1002:D      1      FOR i:=1 TO STLEN(str) DO
1003:D      1      CALL ( iod_wtb , io_tmp_ptr , str[i]);
1004:D      1
1005:D      1      END; { of WITH DO }
1006:D      1
1007:D      1      END; { of writestring }
1008:S
1009:S
1010:S
1011:D      1      PROCEDURE readnumberln
1012:D      1      ( device : type_device ;
1013:D      1      VAR num: REAL);
1014:D      1      VAR io_isc : type_isc;
1015:D      1      BEGIN
1016:D      1      io_isc:=addr_to_talk(device);
1017:D      1      readnumber(io_isc,num);
1018:D      1      IF io_work_char <> io_line_feed THEN
1019:D      1      readuntil(io_line_feed,io_isc);
1020:D      1
1021:D      1      END; { of readnumberln }

```



```

1022:S
1023:S
1024:D      -8 1  PROCEDURE writenumberln
1025:D      -8      ( device : type_device ;
1026:D      -8      value  : REAL );
1027:D      -10 VAR io_isc      : type_isc;
1028:C      BEGIN
1029:C          io_isc:=addr_to_listen(device);
1030:C          writenumber(io_isc,value);
1031:C          writechar(io_isc,io_carriage_rtn);
1032:C          writechar(io_isc,io_line_feed);
1033:C      END; { of writenumberln }
1034:S
1035:S
1036:S
1037:D      -256 1  PROCEDURE writestringln
1038:D      -256      ( device : type_device ;
1039:D      -256      str    : io_STRING );
1040:D      -256 VAR io_isc      : type_isc;
1041:C      BEGIN
1042:C          io_isc:=addr_to_listen(device);
1043:C          writestring(io_isc,str);
1044:C          writechar(io_isc,io_carriage_rtn);
1045:C          writechar(io_isc,io_line_feed);
1046:C      END; { of writestringln }
1047:S
1048:S
1049:S
1050:D      1  PROCEDURE readuntil ( term  : CHAR ;
1051:D          device : type_device );
1052:D      -1  VAR io_work_char: CHAR;
1053:D      -4      io_isc      : type_isc;
1054:C      BEGIN
1055:C          io_isc:=addr_to_talk(device);
1056:S
1057:C      2  WITH isc_table[io_isc].io_drv_ptr^ DO BEGIN
1058:C          3
1059:C              REPEAT
1060:C          4          CALL ( iod_rdb , io_tmp_ptr , io_work_char);
1061:C          4          UNTIL ( io_work_char=term );
1062:C          3
1063:C      END; { of WITH DO BEGIN }
1064:C      3
1065:S
1066:C      2  END; { of readuntil }
1067:S
1068:S
1069:S
1070:D      1  PROCEDURE skipfor ( count : INTEGER ;
1071:D          device : type_device );
1072:D      -4  2  VAR i          : INTEGER;
1073:D      -6      io_isc      : type_isc;
1074:D      -7      io_work_char: CHAR;
1075:C      2  BEGIN
1076:C          io_isc:=addr_to_talk(device);
1077:S
1078:C      2  WITH isc_table[io_isc].io_drv_ptr^ DO BEGIN
1079:C          3
1080:C          isc_table[io_isc] DO BEGIN
1081:C          3          FOR i:=1 TO count DO

```

```

1082:C      4          CALL ( iod_rdb , io_tmp_ptr , io_work_char);
1083:C      3
1084:C      3          END; { of WITH DO BEGIN }
1085:S
1086:C      2  END; { of skipfor }
1087:S
1088:S
1089:S
1090:C      1  END; { of general_2 }

```



```

1091:D      1 $PAGE$
1092:D      1 MODULE general_3 ;
1093:S
1094:S      {
1095:S          date    11/27/81
1096:S          update  07/23/82
1097:S
1098:S          purpose This module contains the LEVEL 3 GENERAL GROUP procedures.
1099:S
1100:D      1      }
1101:S
1102:S
1103:D      1 IMPORT    iodeclarations ;
1104:S
1105:D      1 EXPORT
1106:S
1107:D      1 FUNCTION ierror_message ( ierror : INTEGER ) : io_STRING;
1108:S
1109:D      1 IMPLEMENT
1110:S
1111:D      1 FUNCTION ierror_message ( ierror : INTEGER ) : io_STRING;
1112:D      2 VAR my_msg : io_STRING;
1113:D      2 BEGIN
1114:D      2     my_msg:='zzzz' ;
1115:D      2                                     { 0082 TM 7/23/82 }
1116:D      2     IF ( ierror <= ioe_misc ) AND
1117:D      3     ( ierror >= ioe_no_error )
1118:D      3     THEN BEGIN
1119:D      3         CASE ierror OF
1120:D      4             ioe_no_error      : my_msg := 'no error' ;
1121:D      4             ioe_no_card       : my_msg := 'no card at select code';
1122:D      4             ioe_not_hpib      : my_msg := 'interface should be hpib';
1123:D      4             ioe_not_act       : my_msg := 'not active controller';
1124:D      4             ioe_not_dvc       : my_msg := 'should be device not sc';
1125:D      4             ioe_no_space      : my_msg := 'no space left in buffer';
1126:D      4             ioe_no_data       : my_msg := 'no data left in buffer';
1127:D      4             ioe_bad_tfr       : my_msg := 'improper transfer attempted';
1128:D      4             ioe_isc_busy      : my_msg := 'the select code is busy';
1129:D      4             ioe_buf_busy      : my_msg := 'the buffer is busy';
1130:D      4             ioe_bad_cnt       : my_msg := 'improper transfer count';
1131:D      4             ioe_bad_tmo       : my_msg := 'bad timeout value';
1132:D      4             ioe_no_driver     : my_msg := 'no driver for this card';
1133:D      4             ioe_no_dma        : my_msg := 'no dma';
1134:D      4             ioe_no_word       : my_msg := 'word operations not allowed';
1135:D      4             ioe_not_talk      : my_msg := 'not addressed as talker';
1136:D      4             ioe_not_listen    : my_msg := 'not addressed as listener';
1137:D      4             ioe_timeout       : my_msg := 'a timeout has occurred';
1138:D      4             ioe_not_sctl      : my_msg := 'not system controller';
1139:D      4             ioe_rds_wtc       : my_msg := 'bad status or control';
1140:D      4             ioe_bad_sct       : my_msg := 'bad set/clear/test operation';
1141:D      4             ioe_crd_dwn       : my_msg := 'interface card is dead';
1142:D      4             ioe_eod_seen      : my_msg := 'end/eod has occurred';
1143:D      4             ioe_misc         : my_msg := 'miscellaneous - value of param error';
1144:D      4             END; { of CASE }
1145:D      4         END; { of IF }
1146:D      3     END; { of IF }
1147:S
1148:D      2     IF ( ierror >= ioe_dc_fail ) AND
1149:D      3     ( ierror <= ioe_dc_rval )
1150:D      3     THEN BEGIN

```

```

1151:C      3         CASE ierror OF
1152:S
1153:C      4             ioe_sr_toomany : my_msg := 'too many chars w/o terminator'; { 0364 TM 8/23/82 }
1154:C      4             ioe_dc_fail    : my_msg := 'dc interface failure'; { 0364 TM 8/23/82 }
1155:C      4             ioe_dc_usart    : my_msg := 'USART receive buffer overflow';
1156:C      4             ioe_dc_ovfl      : my_msg := 'receive buffer overflow';
1157:C      4             ioe_dc_clk       : my_msg := 'missing clock';
1158:C      4             ioe_dc_cts       : my_msg := 'CTS false too long';
1159:C      4             ioe_dc_car       : my_msg := 'lost carrier disconnect';
1160:C      4             ioe_dc_act       : my_msg := 'no activity disconnect';
1161:C      4             ioe_dc_conn      : my_msg := 'connection not established';
1162:C      4             ioe_dc_conf      : my_msg := 'bad data bits/parity combination';
1163:C      4             ioe_dc_reg       : my_msg := 'bad status/control register';
1164:C      4             ioe_dc_rval      : my_msg := 'control value out of range';
1165:C      4             END; { of CASE }
1166:C      4         END; { of IF }
1167:C      3     END; { of IF }
1168:S
1169:C      2     IF ioe_result = ioe_sr_fail
1170:C      3     THEN my_msg := 'data-link failure';
1171:C      3                                     { 0364 TM 8/23/82 }
1172:C      2     IF my_msg = 'zzzz' { we don't let sleeping msgs lie }
1173:C      3     THEN CALL(io_error_link , ierror , my_msg );
1174:C      3                                     { 0082 TM 7/23/82 }
1175:C      2     ierror_message := my_msg;
1176:C      2     END; { ierror_message }
1177:C
1178:C
1179:C      1 END; { of general_3 }

```



```

1180:D      1 $PAGES$
1181:D      1 MODULE general_4 ;
1182:D
1183:D      {
1184:D          date 07/17/81
1185:D          update 07/23/82
1186:D
1187:D          purpose This module contains the LEVEL 4 GENERAL GROUP procedures.
1188:D
1189:D      1      }
1190:D
1191:D
1192:D      1 IMPORT      iodeclarations ;
1193:D
1194:D      1 EXPORT
1195:D
1196:D      1
1197:D      2      PROCEDURE abort_transfer
1198:D      2      2      ( VAR b_info: buf_info_type );
1199:D      2      2      FUNCTION transfer_setup
1200:D      2      2      2      ( device      : type_device;
1201:D      2      2      2      t_tfr      : user_tfr_type;
1202:D      2      2      2      t_dir      : dir_of_tfr ;
1203:D      2      2      2      VAR b_info: buf_info_type ;
1204:D      2      2      2      VAR t_cnt : INTEGER )
1205:D      2      2      2      : type_isc ;
1206:D      2      2      2      PROCEDURE transfer
1207:D      2      2      2      2      ( device      : type_device;
1208:D      2      2      2      2      t_tfr      : user_tfr_type;
1209:D      2      2      2      2      t_dir      : dir_of_tfr ;
1210:D      2      2      2      2      VAR b_info: buf_info_type;
1211:D      2      2      2      2      x_count    : INTEGER ) ;
1212:D      2      2      2      PROCEDURE transfer_word
1213:D      2      2      2      2      ( device      : type_device;
1214:D      2      2      2      2      t_tfr      : user_tfr_type;
1215:D      2      2      2      2      t_dir      : dir_of_tfr ;
1216:D      2      2      2      2      VAR b_info: buf_info_type;
1217:D      2      2      2      2      x_count    : INTEGER ) ;
1218:D      2      2      2      PROCEDURE transfer_until
1219:D      2      2      2      2      ( term      : CHAR ;
1220:D      2      2      2      2      device      : type_device;
1221:D      2      2      2      2      t_tfr      : user_tfr_type;
1222:D      2      2      2      2      t_dir      : dir_of_tfr ;
1223:D      2      2      2      2      VAR b_info: buf_info_type ) ;
1224:D      2      2      2      PROCEDURE transfer_end
1225:D      2      2      2      2      ( device      : type_device;
1226:D      2      2      2      2      t_tfr      : user_tfr_type;
1227:D      2      2      2      2      t_dir      : dir_of_tfr ;
1228:D      2      2      2      2      VAR b_info: buf_info_type ) ;
1229:D      2      2      2      PROCEDURE iobuffer ( VAR b_info: buf_info_type ;
1230:D      2      2      2      t_count : INTEGER ) ;
1231:D      2      2      2      PROCEDURE buffer_reset(VAR b_info: buf_info_type ) ;
1232:D      2      2      2      FUNCTION buffer_space(VAR b_info: buf_info_type)
1233:D      2      2      2      : INTEGER;
1234:D      2      2      2      FUNCTION buffer_data( VAR b_info: buf_info_type)
1235:D      2      2      2      : INTEGER;
1236:D      2      2      2      PROCEDURE readbuffer ( VAR b_info: buf_info_type;
1237:D      2      2      2      VAR value : CHAR);
1238:D      2      2      2      PROCEDURE writebuffer( VAR b_info: buf_info_type;
1239:D      2      2      2      value : CHAR);
1240:D      2      2      2      PROCEDURE readbuffer_string

```

```

1240:D      2      ( VAR b_info: buf_info_type;
1241:D      2      VAR str : STRING;
1242:D      2      str_count : INTEGER);
1243:D      2      PROCEDURE writebuffer_string
1244:D      2      2      ( VAR b_info: buf_info_type;
1245:D      2      2      str : io_STRING);
1246:D      2
1247:D      2      FUNCTION buffer_busy( VAR b_info: buf_info_type )
1248:D      2      2      : BOOLEAN;
1249:D      2      2      FUNCTION isc_busy ( isc : type_isc )
1250:D      2      2      : BOOLEAN;
1251:D      2
1252:D      1 IMPLEMENT
1253:D
1254:D
1255:D      1 IMPORT hpib_1 ;
1256:D
1257:D      1 PROCEDURE iobuffer ( VAR b_info: buf_info_type ;
1258:D      2      t_count : INTEGER ) ;
1259:D      2      PROCEDURE NEW $ALIAS 'ASM_NEWBYTES$
***WARNING: (line 1260): External declarations will be treated as global
1260:D      2      (VAR p:ANYPTR;v:INTEGER);EXTERNAL;
1261:D      2      BEGIN
1262:D      2      WITH b_info DO
1263:D      2      BEGIN
1264:D      2      { what about IOBUFFER to a already existant buffer ? }
1265:D      2      { - the space will be thrown away. }
1266:D      2
1267:D      2      NEW(buf_ptr,t_count);
1268:D      2
1269:D      2      act_tfr := no_tfr;
1270:D      2      active_isc:= no_isc;
1271:D      2      buf_size := t_count;
1272:D      2      buf_empty := buf_ptr;
1273:D      2      buf_fill := buf_ptr;
1274:D      2
1275:D      2      drv_imp_ptr := NIL;
1276:D      2      eot_proc.dummy_sl := NIL;
1277:D      2      eot_proc.dummy_pr := NIL;
1278:D      2      eot_parm := NIL;
1279:D      2      dma_priority := FALSE ;
1280:D      2      END; { of WITH DD }
1281:D      2      END; { of iobuffer }
1282:D
1283:D
1284:D
1285:D      1 FUNCTION buffer_data(VAR b_info : buf_info_type )
1286:D      2      : INTEGER;
1287:D      2      BEGIN
1288:D      2      WITH b_info
1289:D      2      DO BEGIN
1290:D      2      buffer_data:=INTEGER(buf_fill)-INTEGER(buf_empty);
1291:D      2      END; { of WITH DD }
1292:D      2      END; { of buffer_data }
1293:D
1294:D
1295:D
1296:D      1 PROCEDURE buffer_reset(VAR b_info: buf_info_type ) ;
1297:D      2      BEGIN
1298:D      2      WITH b_info

```



```
1299:C      3      OO BEGIN
1300:C      3      IF active_isc = no_isc
1301:C      4      THEN BEGIN
1302:C      4          buf_fill:=buf_ptr;
1303:C      4          buf_empty:=buf_ptr;
1304:C      4      END
1305:C      4      ELSE BEGIN
1306:C      4          { error }
1307:C      4          io_escape(ioe_buf_busy,no_isc);
1308:C      4      ENO; { of IF }
1309:C      3      END; { of WITH DO }
1310:C      2      END; { of buffer_reset }
1311:S
1312:S
1313:S
1314:D
1315:D
1316:D      1      FUNCTION buffer_space(VAR b_info: buf_info_type)
1317:D      2          : INTEGER;
1318:D      2      BEGIN
1319:D      2          WITH b_info
1320:D      3          DO BEGIN
1321:D      4              IF ( buffer_data(b_info)=0 ) AND
1322:D      4                  { active_isc = no_isc } THEN buffer_reset(b_info);
1323:D      3              buffer_space:=buf_size+INTEGER(buf_ptr)-INTEGER(buf_fill);
1324:D      3          END; { of WITH DO }
1325:D      2      END; { of buffer_space }
1326:S
1327:D
1328:D      1      PROCEDURE abort_transfer
1329:D      2          ( VAR b_info: buf_info_type );
1330:D      2      BEGIN
1331:D      3          WITH b_info
1332:D      3          DO BEGIN
1333:D      4              IF active_isc <> no_isc
1334:D      4              THEN BEGIN
1335:D      5                  WITH isc_table[active_isc] DO
1336:D      5                      CALL ( io_drv_ptr^.io_d_init ,
1337:D      5                          io_tmp_ptr );
1338:D      5
1339:D      5
1340:D      4              END; { of IF }
1341:D      3          END; { of WITH b_info DO }
1342:D      2      END; { of abort_transfer }
1343:S
1344:S
1345:S
1346:S
1347:S
1348:S
1349:S
1350:D      1      FUNCTION transfer_setup
1351:D      2          ( device      : type_device;
1352:D      2            t_tfr       : user_tfr_type;
1353:D      2            t_dir       : dir_of_tfr;
1354:D      2            VAR b_info: buf_info_type ;
1355:D      2            VAR t_cnt   : INTEGER )
1356:D      2          : type_isc ;
1357:D      -2      VAR io_isc : type_isc;
1358:D      2      BEGIN
```

```
1359:S
1360:C      2      IF device>iomaxisc THEN io_isc := device DIV 100
1361:C      3      ELSE io_isc := device;
1362:S
1363:C      2      IF isc_table[io_isc].io_tmp_ptr = NIL
1364:C      3      THEN io_escape(ioe_no_driver,io_isc);
1365:S
1366:S
1367:D      2      WITH b_info
1368:D      3      DO BEGIN
1369:D      4          { test for tfr count }
1370:D      3          IF t_cnt=0
1371:D      4          THEN BEGIN
1372:D      4              { error }
1373:D      4              io_escape(ioe_bad_cnt,no_isc);
1374:D      4          END;
1375:D      3
1376:D      3          { test for another tfr on this buffer }
1377:D      3          IF active_isc <> no_isc
1378:D      4          THEN BEGIN
1379:D      4              { error }
1380:D      4              io_escape(ioe_buf_busy,no_isc);
1381:D      4          END
1382:D      4          ELSE BEGIN
1383:D      4              IF buffer_data(b_info)=0 THEN buffer_reset(b_info);
1384:D      4          END; { of IF }
1385:D      3
1386:S
1387:S
1388:C      3      { configure card based on direction and check for available space/data }
1389:C      3      IF t_dir=io_memory
1390:C      4      THEN BEGIN
1391:C      5          IF isc_table[io_isc].io_tmp_ptr^.in_bufptr <> NIL
1392:C      5          THEN BEGIN
1393:C      5              { error }
1394:C      5              io_escape(ioe_isc_busy,io_isc);
1395:C      5          END; { of IF }
1396:C      5          IF buffer_space(b_info)<t_cnt
1397:C      5          THEN BEGIN
1398:C      5              { error }
1399:C      5              io_escape(ioe_no_space,io_isc);
1400:C      5          END; { of IF }
1401:C      4          io_isc:=addr_to_talk(device);
1402:C      4          isc_table[io_isc].io_tmp_ptr^.in_bufptr := ADDR( b_info );
1403:C      4          END
1404:C      4          ELSE BEGIN
1405:C      5              IF isc_table[io_isc].io_tmp_ptr^.out_bufptr <> NIL
1406:C      5              THEN BEGIN
1407:C      5                  { error }
1408:C      5                  io_escape(ioe_isc_busy,io_isc);
1409:C      5              END; { of IF }
1410:C      4              IF buffer_data(b_info)<t_cnt
1411:C      4              THEN BEGIN
1412:C      4                  { error }
1413:C      4                  io_escape(ioe_no_data,io_isc);
1414:C      4              END; { of IF }
1415:C      4              io_isc:=addr_to_listen(device);
1416:C      4              isc_table[io_isc].io_tmp_ptr^.out_bufptr := ADDR( b_info );
1417:C      4          END; { of IF }
1418:S
```



```

1419:C      3      drv_tmp_ptr:= isc_table[io_isc].io_tmp_ptr;
1420:C      3      act_tfr := no_tfr;
1421:C      3      usr_tfr := t_tfr;
1422:C      3      b_w_mode := FALSE;           { byte mode }
1423:C      3      end_mode := FALSE;           { no EOI }
1424:C      3      direction := t_dir;
1425:C      3      term_char := -1;             { no termination character }
1426:C      3      term_count := t_cnt;
1427:S
1428:C      3      ENO; { of WITH b_info 00 }
1429:S
1430:C      2      transfer_setup := io_isc;
1431:S
1432:C      2      ENO; { of transfer_setup }
1433:S
1434:S
1435:S
1436:O      1  PROCEDURE transfer ( device : type_device;
1437:O      2      t_tfr : user_tfr_type;
1438:O      2      t_dir : dir_of_tfr;
1439:O      2      VAR b_info: buf_info_type ;
1440:O      2      x_count : INTEGER ) ;
1441:D      -2  VAR io_isc : type_isc;
1442:O      -6  t_count : INTEGER;
1443:C      2  BEGIN
1444:C      2      t_count:=x_count;
1445:C      2      io_isc:=transfer_setup(device,t_tfr,t_dir,b_info,t_count);
1446:S
1447:C      2      { transfer temporary was set up in transfer_setup }
1448:S
1449:C      2      WITH isc_table[io_isc]
1450:C      2      DO CALL ( io_drv_ptr^.iod_tfr ,
1451:C      3      isc_table[io_isc].io_tmp_ptr,
1452:C      3      ADDR(b_info) );
1453:S
1454:C      2      ENO; { of transfer }
1455:S
1456:S
1457:S
1458:S
1459:O      1  PROCEDURE transfer_word
1460:O      2      ( device : type_device;
1461:O      2      t_tfr : user_tfr_type;
1462:O      2      t_dir : dir_of_tfr;
1463:O      2      VAR b_info: buf_info_type ;
1464:O      2      x_count : INTEGER ) ;
1465:D      -2  VAR io_isc : type_isc;
1466:O      -6  t_count : INTEGER;
1467:C      2  BEGIN
1468:C      2      t_count:=x_count;
1469:C      2      io_isc:=transfer_setup(device,t_tfr,t_dir,b_info,t_count);
1470:S
1471:C      2      { fix up transfer temporary }
1472:C      2      b_info.b_w_mode := TRUE;           { word mode }
1473:S
1474:C      2      WITH isc_table[io_isc]
1475:C      2      DO CALL ( io_drv_ptr^.iod_tfr ,
1476:C      3      isc_table[io_isc].io_tmp_ptr,
1477:C      3      ADDR(b_info) );
1478:S

```

```

1479:C      2      ENO; { of transfer_word }
1480:S
1481:S
1482:S
1483:S
1484:O      1  PROCEDURE transfer_until
1485:O      2      ( term : CHAR ;
1486:O      2      device : type_device;
1487:O      2      t_tfr : user_tfr_type;
1488:O      2      t_dir : dir_of_tfr;
1489:O      2      VAR b_info: buf_info_type ) ;
1490:D      -2  VAR io_isc : type_isc;
1491:O      -6  t_count : INTEGER;
1492:C      2  BEGIN
1493:C      2      t_count := buffer_space(b_info);
1494:C      2      io_isc:=transfer_setup(device,t_tfr,t_dir,b_info,t_count);
1495:S
1496:C      2      IF t_dir = from_memory
1497:C      2      THEN BEGIN
1498:C      3      { error }
1499:C      3      io_escape(ioe_bad_tfr,io_isc);
1500:C      3      ENO;
1501:S
1502:C      2      WITH b_info DO BEGIN
1503:C      3      { fix up transfer temporary }
1504:C      3      term_char := ord(term);           { termination character }
1505:C      3
1506:C      3      { check for transfer type cases }
1507:C      3      IF t_tfr = serial_FASTEST THEN usr_tfr := serial_FHS;
1508:C      3      IF t_tfr = overlap_FASTEST THEN usr_tfr := overlap_FHS;
1509:C      3      IF t_tfr = OVERLAP THEN usr_tfr := overlap_INTR;
1510:C      3      IF ( t_tfr = serial_DMA ) OR ( t_tfr = overlap_DMA )
1511:C      4      THEN BEGIN
1512:C      4      { error }
1513:C      4      io_escape(ioe_bad_tfr,io_isc);
1514:C      4      ENO; { of IF }
1515:C      4
1516:S
1517:C      3      END; { of WITH b_info 00 BEGIN }
1518:S
1519:S
1520:C      2      WITH isc_table[io_isc]
1521:C      3      DO CALL ( io_drv_ptr^.iod_tfr ,
1522:C      3      isc_table[io_isc].io_tmp_ptr,
1523:C      3      ADDR(b_info) );
1524:S
1525:C      2      ENO; { of transfer_until }
1526:S
1527:S
1528:S
1529:O      1  PROCEDURE transfer_end(device : type_device;
1530:O      2      t_tfr : user_tfr_type;
1531:O      2      t_dir : dir_of_tfr;
1532:O      2      VAR b_info: buf_info_type ) ;
1533:D      -2  VAR io_isc : type_isc;
1534:O      -6  t_count : INTEGER;
1535:C      2  BEGIN
1536:C      2      IF t_dir=from_memory
1537:C      2      THEN BEGIN
1538:C      3      t_count := buffer_data(b_info);

```



```

1539:C      3      END
1540:C      3      ELSE BEGIN
1541:C      3          t_count := buffer_space(b_info);
1542:C      3      END; { of IF }
1543:C      2      io_isc:=transfer_setup(device,t_tfr,t_dir,b_info,t_count);
1544:S
1545:C      2      { fix up transfer temporary }
1546:C      2      b_info.end_mode := TRUE;          { EDI }
1547:S
1548:C      2      WITH isc_table[io_isc]
1549:C      3      DO CALL ( io_drv_ptr^.iod_tfr ,
1550:C      3          isc_table[io_isc].io_imp_ptr,
1551:C      3          ADDR(b_info));
1552:S
1553:C      2      END; { of transfer_end }
1554:S
1555:S
1556:S
1557:D      1      PPOCEDURE readbuffer ( VAR b_info: buf_info_type;
1558:D      2          VAR p : ^CHAR;          VAR value : CHAR);
1559:D      -4      BEGIN
1560:C      2      IF buffer_data(b_info)<1
1561:C      3      THEN BEGIN
1562:C      3          { error }
1563:C      3          io_escape(ioe_no_data,no_isc);
1564:C      3      END
1565:C      3      ELSE BEGIN
1566:C      3          WITH b_info
1567:C      4          DO BEGIN
1568:C      4              IF ( active_isc <> no_isc ) AND
1569:C      4                  ( direction = from_memory )
1570:C      5              THEN BEGIN
1571:C      5                  { error }
1572:C      5                  io_escape( no_isc , ioe_buf_busy );
1573:C      5              END; { of IF }
1574:C      4              p:=ANYPTR(buf_empty);
1575:C      4              value:=p;
1576:C      4              buf_empty:=ANYPTR(INTEGER(buf_empty)+1);
1577:C      4          END; { of WITH b_info DO }
1578:C      3      END; { of IF }
1579:C      2      END; { of readbuffer }
1580:S
1581:S
1582:S
1583:S
1584:D      1      PPOCEDURE writebuffer( VAR b_info: buf_info_type;
1585:D      2          value : CHAR);
1586:D      -4      VAR p : ^CHAR;
1587:D      2      BEGIN
1588:C      2      IF buffer_space(b_info)<1
1589:C      3      THEN BEGIN
1590:C      3          { error }
1591:C      3          io_escape(ioe_no_space,no_isc);
1592:C      3      END
1593:C      3      ELSE BEGIN
1594:C      3          WITH b_info
1595:C      4          DO BEGIN
1596:C      4              IF ( active_isc <> no_isc ) AND
1597:C      5              ( direction = to_memory )
1598:C      5              THEN BEGIN

```

```

1599:C      5          { error }
1600:C      5          io_escape( no_isc , ioe_buf_busy );
1601:C      5      END; { of IF }
1602:C      4      p:=buf_fill;
1603:C      4      p:=value;
1604:C      4      buf_fill:=ANYPTR(INTEGER(buf_fill)+1);
1605:C      4      END; { of WITH b_info DO }
1606:C      3      END; { of IF }
1607:C      2      END; { of writebuffer }
1608:S
1609:S
1610:S
1611:D      1      PROCEDURE readbuffer_string
1612:D      2          ( VAR b_info: buf_info_type;
1613:D      2              VAR str : STRNG;
1614:D      2              str_count : INTEGER);
1615:D      -4      VAR i : INTEGER;
1616:D      2      BEGIN
1617:C      2      IF STRMAX(str) < str_count
1618:C      3      THEN BEGIN
1619:C      3          { error - string too small }
1620:C      3          io_escape(ioe_misc,no_isc);
1621:C      3      END;
1622:C      2      SETSTRLEN(str,str_count);          { so I can put chars into empty string }
1623:C      2      IF buffer_data(b_info)<str_count
1624:C      3      THEN BEGIN
1625:C      3          { error - not enough data in buffer }
1626:C      3          io_escape(ioe_no_data,no_isc);
1627:C      3      END
1628:C      3      ELSE BEGIN
1629:C      3          FOR i:=1 TO str_count
1630:C      4          DO BEGIN
1631:C      4              readbuffer(b_info,str[i]);
1632:C      4          END; { of FOR BEGIN }
1633:C      3          SETSTRLEN(str,str_count);
1634:C      3      END; { of IF }
1635:C      2      END; { of readbuffer_string }
1636:S
1637:S
1638:S
1639:D      -256 1      PROCEDURE writebuffer_string
1640:D      2          ( VAR b_info: buf_info_type;
1641:D      2              str : io_STRING);
1642:D      -260 2      VAR i : INTEGER;
1643:D      2      BEGIN
1644:C      2      IF buffer_space(b_info)<STRLEN(str)
1645:C      3      THEN BEGIN
1646:C      3          { error }
1647:C      3          io_escape(ioe_no_space,no_isc);
1648:C      3      END
1649:C      3      ELSE BEGIN
1650:C      3          FOR i:=1 TO STRLEN(str)
1651:C      4          DO BEGIN
1652:C      4              writebuffer(b_info,str[i]);
1653:C      4          END; { of FOR BEGIN }
1654:C      3      END; { of IF }
1655:C      2      END; { of writebuffer_string }
1656:S
1657:S
1658:D      1      FUNCTION buffer_busy( VAR b_info: buf_info_type )

```



```

1658:O      2      : BOOLEAN;      { 0083 TM 7/23/82 }
1659:C      2      BEGIN      { 0083 TM 7/23/82 }
1660:C      2      WITH b_info DO BEGIN      { 0083 TM 7/23/82 }
1661:C      3      IF active_isc <> no_isc THEN buffer_busy := TRUE;      { 0083 TM 7/23/82 }
1662:C      3      IF active_isc <> no_isc THEN buffer_busy := TRUE;      { 0083 TM 7/23/82 }
1663:C      4      ENO; { of WITH DO BEGIN }      { 0083 TM 7/23/82 }
1664:C      3      ENO; { of WITH DO BEGIN }      { 0083 TM 7/23/82 }
1665:C      2      END; { of buffer_busy }      { 0083 TM 7/23/82 }
1666:S
1667:D      1      FUNCTION isc_busy ( isc : type_isc )      { 0083 TM 7/23/82 }
1668:D      1      : BOOLEAN;      { 0083 TM 7/23/82 }
1669:C      2      BEGIN      { 0083 TM 7/23/82 }
1670:C      2      WITH isc_table[isc].io_tmp_ptr^ DO BEGIN      { 0083 TM 7/23/82 }
1671:C      3      IF ( in_bufptr <> NIL ) OR      { 0083 TM 7/23/82 }
1672:C      4      ( out_bufptr <> NIL ) THEN isc_busy := TRUE;      { 0083 TM 7/23/82 }
1673:C      4      ELSE isc_busy := FALSE;      { 0083 TM 7/23/82 }
1674:C      3      ENO; { of WITH DO BEGIN }      { 0083 TM 7/23/82 }
1675:C      2      END; { of isc_busy }      { 0083 TM 7/23/82 }
1676:S
1677:S
1678:S
1679:C      1      END; { of general_4 }

```

```

1680:D      1      $PAGE$
1681:D      1      {*****}
1682:D      1      {
1683:D      1      {
1684:D      1      {      HPPIB GROUP
1685:D      1      {
1686:D      1      {
1687:D      1      {*****}
1688:S
1689:S
1690:S
1691:S
1692:O      1      MODULE hpib_0 ;
1693:S
1694:S      {
1695:S      {      date      07/17/81
1696:S      {      update    09/17/81
1697:S      {
1698:S      {      purpose This module contains the LEVEL 0 HPPIB GROUP procedures.
1699:S      {
1700:O      1      }
1701:S
1702:S
1703:D      1      IMPORT      iodeclarations ;
1704:S
1705:S
1706:D      1      EXPORT
1707:S
1708:S
1709:D      1      PROCEDURE set_hpib      ( select_code : type_isc ;
1710:D      2      line      : type_hpib_line);
1711:D      1      PROCEDURE clear_hpib      ( select_code : type_isc ;
1712:D      2      line      : type_hpib_line);
1713:D      2      FUNCTION hpib_line      ( select_code : type_isc ;
1714:D      2      line      : type_hpib_line)
1715:D      2      : BOOLEAN;
1716:S
1717:D      1      IMPLEMENT
1718:S
1719:S
1720:S
1721:S
1722:D      1      PROCEDURE set_hpib      ( select_code : type_isc ;
1723:D      2      line      : type_hpib_line);
1724:C      2      BEGIN
1725:C      2      WITH isc_table[select_code] DO
1726:C      3      CALL(io_drv_ptr^.iod_set,
1727:C      3      io_tmp_ptr,
1728:C      3      ORD(line));
1729:C      2      ENO;
1730:S
1731:S
1732:S
1733:D      1      PROCEDURE clear_hpib      ( select_code : type_isc ;
1734:D      2      line      : type_hpib_line);
1735:C      2      BEGIN
1736:C      2      WITH isc_table[select_code] DO
1737:C      3      CALL(io_drv_ptr^.iod_clr,
1738:C      3      io_tmp_ptr,
1739:C      3      ORD(line));

```



```

1740:C      2  END;
1741:S
1742:S
1743:S
1744:D      1  FUNCTION hpib_line ( select_code : type_isc ;
1745:D      2      line      : type_hpib_line)
1746:D      3      : BOOLEAN;
1747:D      -1  2  VAR my_boolean : BOOLEAN;
1748:C      2  BEGIN
1749:C      3  WITH isc_table[select_code] DO
1750:C      3      CALL(io_drv_ptr^.iod_test,
1751:C      3      io_tmp_ptr,
1752:C      3      ORD(line),
1753:C      3      my_boolean);
1754:C      2  hpib_line:=my_boolean;
1755:C      2  END;
1756:S
1757:S
1758:S
1759:C      1  END;  ( of hpib_0 )

```

```

1760:D      1  $PAGE$
1761:D      1  MODULE hpib_2 ;
1762:S
1763:S      {
1764:S          date   07/17/81
1765:S          update  03/09/83
1766:S
1767:S          purpose This module contains the LEVEL
1768:S                  2 HPIB GROUP procedures.
1769:S      }
1770:D      1
1771:S
1772:S
1773:D      1  IMPORT iodeclarations ;
1774:S
1775:D      1  EXPORT
1776:S
1777:S
1778:D      1  PROCEDURE abort_hpib
1779:D      2      ( select_code : type_isc);
1780:D      1  PROCEDURE clear   ( device   : type_device);
1781:D      1  PROCEDURE listen  ( select_code : type_isc ;
1782:D      2      address   : type_hpib_addr );
1783:D      1  PROCEDURE local   ( device   : type_device);
1784:D      1  PROCEDURE local_lockout
1785:D      2      ( select_code : type_isc);
1786:D      1  PROCEDURE pass_control
1787:D      2      ( device   : type_device);
1788:D      1  PROCEDURE ppoll_configure
1789:D      2      ( device   : type_device;
1790:D      2      mask     : INTEGER );
1791:D      1  PROCEDURE ppoll_unconfigure
1792:D      2      ( device   : type_device);
1793:D      1  PROCEDURE remote  ( device   : type_device);
1794:D      1  PROCEDURE secondary
1795:D      2      ( select_code : type_isc ;
1796:D      2      address   : type_hpib_addr );
1797:D      1  PROCEDURE talk    ( select_code : type_isc ;
1798:D      2      address   : type_hpib_addr );
1799:D      1  PROCEDURE trigger ( device   : type_device);
1800:D      1  PROCEDURE unlisten( select_code : type_isc );
1801:D      1  PROCEDURE untalk  ( select_code : type_isc );
1802:S
1803:S
1804:D      1  IMPLEMENT
1805:S
1806:D      1  IMPORT hpib_0 ;
1807:D      1  hpib_1 ;
1808:S
1809:S
1810:D      1  PROCEDURE abort_hpib
1811:D      2      ( select_code : type_isc);
1812:C      2  BEGIN
1813:S      2      ( what about active tfrs ? )
1814:C
1815:S
1816:C      2  IF system_controller(select_code)
1817:C      3  THEN BEGIN
1818:C      3      set_hpib(select_code,ifc_line);
1819:C      3      set_hpib(select_code,ren_line);

```



```
1820:C      3      clear_hplib(select_code,ifc_line);
1821:C      3      clear_hplib(select_code,atn_line);      { all done by ifc }
1822:C      3      END
1823:C      3      ELSE BEGIN
1824:C      3      IF active_controller(select_code)
1825:C      4      THEN BEGIN
1826:C      4      send_command(select_code,
1827:C      4      CHR(talk_constant+my_address(select_code)));
1828:C      4      send_command(select_code,?');
1829:C      4      clear_hplib(select_code,atn_line);
1830:C      4      END
1831:C      4      ELSE BEGIN
1832:C      4      { do nothing }
1833:C      4      END; { of IF }
1834:C      3      END; { of IF }
1835:C      2      END; { of abort_hplib }
1836:S
1837:S
1838:D      1      PROCEDURE clear ( device      : type_device);
1839:D      2      VAR io_isc      : type_isc;
1840:C      2      BEGIN
1841:C      2      io_isc:=set_to_listen(device);
1842:C      2      IF device>iomaxisc
1843:C      3      THEN BEGIN
1844:C      3      send_command(io_isc,scd_message);
1845:C      3      END
1846:C      3      ELSE BEGIN
1847:C      3      send_command(io_isc,dcl_message);
1848:C      3      END; { of IF }
1849:C      2      END; { of clear }
1850:S
1851:S
1852:S
1853:D      1      PROCEDURE listen ( select_code : type_isc ;
1854:D      2      address      : type_hplib_addr );
1855:C      2      BEGIN
1856:C      2      send_command(select_code,CHR(listen_constant+address));
1857:C      2      END; { of listen }
1858:S
1859:S
1860:S
1861:D      1      PROCEDURE local ( device      : type_device);
1862:D      2      VAR io_isc      : type_isc;
1863:C      2      BEGIN
1864:C      2      IF device>iomaxisc
1865:C      3      THEN BEGIN
1866:C      3      io_isc:=set_to_listen(device);      { BUG 1251 TM 1/8/82 }
1867:C      3      send_command(io_isc,gtl_message);
1868:C      3      END
1869:C      3      ELSE BEGIN
1870:C      3      io_isc := device;      { BUG 1251 TM 1/8/82 }
1871:C      3      IF system_controller(io_isc)      { BUG jsjs TM 3/9/83 }
1872:C      4      THEN BEGIN
1873:C      4      { system controller - drop REN }      { BUG jsjs TM 3/9/83 }
1874:C      4      clear_hplib(io_isc,ren_line);
1875:C      4      IF active_controller(io_isc)      { BUG 1251 TM 1/26/82 }
1876:C      5      THEN clear_hplib(io_isc,atn_line);      { BUG 1251 TM 1/26/82 }
1877:C      4      END      { BUG jsjs TM 3/9/83 }
1878:C      4      ELSE BEGIN      { BUG jsjs TM 3/9/83 }
1879:C      4      { not system controller - send GTL }      { BUG jsjs TM 3/9/83 }
```

```
1880:C      4      send_command(io_isc,gtl_message);      { BUG jsjs TM 3/9/83 }
1881:C      4      END; { of IF }      { BUG jsjs TM 3/9/83 }
1882:C      3      END; { of IF }
1883:C      2      END; { of local }
1884:S
1885:S
1886:S
1887:S
1888:D      1      PROCEDURE local_lockout
1889:D      2      ( select_code : type_isc);
1890:C      2      BEGIN
1891:C      2      send_command(select_code,llo_message);
1892:C      2      END; { of local_lockout }
1893:S
1894:S
1895:S
1896:D      1      PROCEDURE pass_control
1897:D      2      ( device      : type_device);
1898:D      2      VAR io_isc      : type_isc;
1899:C      2      BEGIN
1900:C      2      IF device>iomaxisc      { BUG 1258 TM 1/8/82 }
1901:C      3      THEN BEGIN      { BUG 1258 TM 1/8/82 }
1902:C      3      io_isc := device DIV 100;      { BUG 1258 TM 1/8/82 }
1903:C      3      send_command(io_isc,unl_message);      { BUG 1258 TM 1/8/82 }
1904:C      3      send_command(io_isc,
1905:C      3      chr((device MOD 100)+talk_constant));      { BUG 1258 TM 1/8/82 }
1906:C      3      END      { BUG 1258 TM 1/8/82 }
1907:C      3      ELSE BEGIN      { BUG 1258 TM 1/8/82 }
1908:C      3      io_isc := set_to_talk(device);      { BUG 1258 TM 1/8/82 }
1909:C      3      END; { of IF device }      { BUG 1258 TM 1/8/82 }
1910:C      2      send_command(io_isc,tct_message);
1911:C      2      END;
1912:S
1913:S
1914:S
1915:D      1      PROCEDURE ppoll_configure
1916:D      2      ( device      : type_device;
1917:D      2      mask      : INTEGER );
1918:D      2      VAR io_isc      : type_isc;
1919:C      2      BEGIN
1920:C      2      io_isc:=set_to_listen(device);
1921:C      2      IF io_isc=device
1922:C      3      THEN BEGIN
1923:C      3      { error }
1924:C      3      io_escape(ioe_not_dvc,io_isc);
1925:C      3      END
1926:C      3      ELSE BEGIN
1927:C      3      send_command(io_isc,ppc_message);
1928:C      3      send_command(io_isc,CHR(ord(ppc_message)+(mask MOD 16)));
1929:C      3      END; { of IF }
1930:C      2      END; { of ppoll_configure }
1931:S
1932:S
1933:S
1934:D      1      PROCEDURE ppoll_unconfigure
1935:D      2      ( device      : type_device);
1936:D      2      VAR io_isc      : type_isc;
1937:C      2      BEGIN
1938:C      2      io_isc:=set_to_listen(device);
1939:C      2      IF device>iomaxisc
```



```

1940:C      3      THEN BEGIN
1941:C      3          send_command(io_isc,ppc_message);
1942:C      3          send_command(io_isc,ppd_message);
1943:C      3      END
1944:C      3      ELSE BEGIN
1945:C      3          send_command(io_isc,ppu_message);
1946:C      3      END; { of IF }
1947:C      2      END; { of ppoll_unconfigure }
1948:S
1949:S
1950:S
1951:D      1      PROCEDURE remote ( device      : type_device);
1952:D      -2      VAR io_isc      : type_isc;
1953:C      2      BEGIN
1954:C      2          IF device>iomaxisc
1955:C      3          THEN BEGIN
1956:C      3              io_isc:=device DIV 100;
1957:C      3              IF NOT system_controller(io_isc)
1958:C      4                  THEN io_escape(ioe_not_sctl,io_isc);
1959:C      4                  set_hpib(io_isc,ren_line);
1960:C      3              io_isc:=set_to_listen(device);
1961:C      3          END
1962:C      3          ELSE BEGIN
1963:C      3              io_isc := device;
1964:C      3              IF NOT system_controller(io_isc)
1965:C      4                  THEN io_escape(ioe_not_sctl,io_isc);
1966:C      4                  set_hpib(io_isc,ren_line);
1967:C      3              END; { of IF }
1968:C      2      END; { of remote }
1969:S
1970:S
1971:S
1972:D      1      PROCEDURE secondary
1973:D      2          ( select_code : type_isc ;
1974:D      2            address      : type_hpib_addr );
1975:C      2      BEGIN
1976:C      2          send_command(select_code,CHR(address+96));
1977:C      2      END; { of secondary }
1978:S
1979:S
1980:S
1981:D      1      PROCEDURE talk      ( select_code : type_isc ;
1982:D      2            address      : type_hpib_addr );
1983:C      2      BEGIN
1984:C      2          send_command(select_code,CHR(address+talk_constant));
1985:C      2      END; { of talk }
1986:S
1987:S
1988:S
1989:D      1      PROCEDURE trigger ( device      : type_device);
1990:C      2      BEGIN
1991:C      2          send_command(set_to_listen(device),get_message);
1992:C      2      END; { of trigger }
1993:S
1994:S
1995:D      1      PROCEDURE unlisten( select_code : type_isc );
1996:C      2      BEGIN
1997:C      2          send_command(select_code,unl_message);
1998:C      2      END; { of unlisten }
1999:S

```

```

2000:S
2001:D      1      PROCEDURE untalk ( select_code : type_isc );
2002:C      2      BEGIN
2003:C      2          send_command(select_code,unt_message);
2004:C      2      END; { of untalk }
2005:S
2006:S
2007:S
2008:S
2009:C      1      END;      { of hpib_2 }

```



```

2010:D      1 $PAGE$
2011:D      1 MODULE hpib_3 ;
2012:S
2013:S      {
2014:S          date 07/17/81
2015:S          update 01/08/82
2016:S
2017:S          purpose This module contains the LEVEL
2018:S                    3 HP18 GROUP procedures.
2019:S      }
2020:D      1
2021:S
2022:S
2023:D      1 IMPORT iodeclarations ;
2024:S
2025:D      1 EXPORT
2026:S
2027:S
2028:D      1 FUNCTION requested
2029:D          ( select_code : type_isc )
2030:D          : BOOLEAN ;
2031:D      1 FUNCTION ppoll
2032:D          ( select_code : type_isc )
2033:D          : INTEGER ;
2034:D      1 FUNCTION spoll
2035:D          ( device : type_device )
2036:D          : INTEGER ;
2037:D
2038:D      1 PROCEDURE request_service
2039:D          ( select_code : type_isc ;
2040:D            response : INTEGER ) ;
2041:D      1 FUNCTION listener
2042:D          ( select_code : type_isc )
2043:D          : BOOLEAN ;
2044:D      1 FUNCTION talker
2045:D          ( select_code : type_isc )
2046:D          : BOOLEAN ;
2047:D      1 FUNCTION remotod
2048:D          ( select_code : type_isc )
2049:D          : BOOLEAN ;
2050:D
2051:D      1 IMPLEMENT
2052:S
2053:D      1 IMPORT iocomasm ,
2054:D            general_0 ,
2055:D            general_1 ,
2056:D            hpib_0 ,
2057:D            hpib_1 ;
2058:S
2059:S
2060:S
2061:S
2062:D      1 FUNCTION requested
2063:D          ( select_code : type_isc )
2064:D          : BOOLEAN ;
2065:D      2 BEGIN
2066:D          IF active_controller(select_code)
2067:D          THEN BEGIN
2068:D              requested:=hpib_line(select_code,srq_line);
2069:D          ENO

```

```

2070:C      3 ELSE BEGIN
2071:C      3 { error - not active controller when look at srq }
2072:C      3 { escape(ioe_not_act,select_code);
2073:C      3 ENO; { of IF }
2074:C      2 ENO; { of requested }
2075:S
2076:S
2077:S
2078:S
2079:S
2080:D      1 FUNCTION ppoll ( select_code : type_isc )
2081:D          : INTEGER ;
2082:D      -1 2 VAR my_byte : CHAR;
2083:D      2 BEGIN
2084:D          WITH isc_table[select_code] DO
2085:D              CALL(io_drv_ptr^.iod_ppoll,
2086:D                    io_imp_ptr,
2087:D                    my_byte);
2088:D          ppoll:=ORD(my_byte);
2089:D      2 END; { of ppoll }
2090:S
2091:S
2092:S
2093:S
2094:D      1 FUNCTION spoll ( device : type_device )
2095:D          : INTEGER ;
2096:D      -2 2 VAR io_isc : type_isc;
2097:D      -3 2 io_work_char: CHAR;
2098:D      2 BEGIN
2099:D          io_isc:=set_to_talk(device);
2100:D          send_command(io_isc,spe_message);
2101:D          readchar(io_isc,io_work_char);
2102:D          send_command(io_isc,spd_message);
2103:D          send_command(io_isc,unt_message);
2104:D          spoll:=ord(io_work_char);
2105:D      2 END; { of spoll }
2106:S
2107:S
2108:S
2109:S
2110:S
2111:S
2112:S
2113:S
2114:S
2115:S
2116:D      1 PROCEDURE request_service
2117:D          ( select_code : type_isc ;
2118:D            response : INTEGER ) ;
2119:D      2 BEGIN
2120:D          IF isc_table[select_code].card_type=hpib_card
2121:D          THEN BEGIN
2122:D              IF NOT active_controller(select_code) { BUG 1250 TH 1/8/82 }
2123:D              THEN iocontrol(select_code,1,response) { BUG 1250 TH 1/8/82 }
2124:D              ELSE io_escape(ioe_misc,select_code); { BUG 1250 TH 1/8/82 }
2125:D          ENO
2126:D          ELSE BEGIN
2127:D              { error }
2128:D              io_escape(ioe_not_hpib,select_code);
2129:D          ENO; { of IF }

```



```

2130:C      2   END; { of request_service }
2131:S
2132:S
2133:S
2134:S
2135:D      1   FUNCTION listener( select_code : type_isc )
2136:C      2           : BOOLEAN;
2137:C      2   BEGIN
2138:C      2       listener:=bit_set(iostatus(select_code,6),10);
2139:C      2   END; { of listener }
2140:S
2141:S
2142:S
2143:S
2144:S
2145:D      1   FUNCTION talker { select_code : type_isc )
2146:D      2           : BOOLEAN ;
2147:C      2   BEGIN
2148:C      2       talker:=bit_set(iostatus(select_code,6),9);
2149:C      2   END; { of talker }
2150:S
2151:S
2152:S
2153:S
2154:D      1   FUNCTION remotod { select_code : type_isc )
2155:D      2           : BOOLEAN ;
2156:C      2   BEGIN
2157:C      2       remotod:=bit_set(iostatus(select_code,6),15);
2158:C      2   END; { of remotod }
2159:S
2160:S
2161:S
2162:S
2163:D      1   FUNCTION locked_out
2164:D      2           { select_code : type_isc )
2165:D      2           : BOOLEAN ;
2166:C      2   BEGIN
2167:C      2       locked_out:=bit_set(iostatus(select_code,6),14);
2168:C      2   END;
2169:S
2170:S
2171:S
2172:S
2173:C      1   ENO.      { of hpib 3 }

```

```

2174:D 1 $PAGE$
2175:D 1 *****
2176:D 1 *
2177:D 1 *
2178:D 1 * SERIAL GROUP
2179:D 1 *
2180:D 1 *
2181:D 1 *****
2182:D 1 *
2183:D 1 *
2184:D 1 * The 98626 code in the serial_0 and serial_3 modules has NOT
2185:D 1 * been tested and is included in the hopes that it is correct
2186:D 1 * and that someone will do the 98626 card drivers sometime.
2187:D 1 *
2188:D 1 * There is a good chance that the 98626 will require a re-
2189:D 1 * release of the IOLIB:IOLIB file ( serial modules only ).
2190:D 1 *
2191:D 1 { *****
2192:S
2193:S
2194:S
2195:S
2196:D 1 MODULE serial_0 ;
2197:S
2198:S {
2199:S date 07/22/81
2200:S update 11/06/81
2201:S
2202:S purpose This module contains the LEVEL
2203:S 0 SERIAL GROUP procedures.
2204:S
2205:D 1 }
2206:S
2207:S
2208:D 1 IMPORT iodeclarations ;
2209:S
2210:D 1 EXPORT
2211:S
2212:S
2213:S
2214:S
2215:D 1 PROCEDURE set_serial ( select_code : type_isc ;
2216:D 2 line : type_serial_line);
2217:D 1 PROCEDURE clear_serial( select_code : type_isc ;
2218:D 2 line : type_serial_line);
2219:D 1 FUNCTION serial_line ( select_code : type_isc ;
2220:D 2 line : type_serial_line)
2221:D 2 : 800LEAN;
2222:S
2223:S
2224:D 1 IMPLEMENT
2225:S
2226:D 1 IMPORT iocomasm ,
2227:D 1 general_0 ;
2228:S
2229:S
2230:S
2231:D 1 PROCEDURE set_serial ( select_code : type_isc ;
2232:D 2 line : type_serial_line);
2233:D -4 2 VAR mybit : INTEGER;

```



```

2234:D  -8 2 dummy : INTEGER;
2235:C  2 BEGIN
2236:C  2 WITH isc_table[select_code] DO BEGIN
2237:S
2238:C  3 IF card_type <> serial_card THEN io_escape(ioe_bad_sct,select_code);
2239:C  3
2240:C  3 IF (isc_table[select_code].card_id = hp98628_async)
2241:C  4 THEN BEGIN
2242:C  4 CASE line OF
2243:S
2244:C  5 rts_line: mybit := 1;
2245:S
2246:C  5 dtr_line: mybit := 2;
2247:C  5
2248:C  5 drs_line: mybit := 4;
2249:S
2250:C  5 OTHERWISE io_escape(ioe_bad_sct,select_code);
2251:S
2252:C  5 END; { of CASE line }
2253:C  4 dummy := iostatus(select_code,8);
2254:C  4 dummy := binior(dummy,mybit);
2255:C  4 iocontrol(select_code,8*256,dummy);
2256:S
2257:C  4 END
2258:C  4 ELSE BEGIN
2259:C  4
2260:C  4 IF (card_id = hp98626) OR (card_id = hp98644)
2261:C  5 THEN BEGIN
2262:C  5 CASE line OF
2263:S
2264:C  6 rts_line: mybit := 2;
2265:S
2266:C  6 dtr_line: mybit := 1;
2267:S
2268:C  6 drs_line: mybit := 8;
2269:S
2270:C  6 OTHERWISE io_escape(ioe_bad_sct,select_code);
2271:S
2272:C  6 END; { of CASE line }
2273:S
2274:C  5 dummy := iostatus(select_code,5);
2275:C  5 dummy := binior(dummy,mybit);
2276:C  5 iocontrol(select_code,5,dummy);
2277:S
2278:C  5 END
2279:C  5 ELSE BEGIN
2280:C  5 CALL ( io_drv_ptr^.iod_set ,
2281:C  5 io_imp_ptr ,
2282:C  5 ORD(line) );
2283:C  5 END; { of IF }
2284:S
2285:C  4 END; { of IF }
2286:S
2287:C  3 END; { of WITH isc_table BEGIN }
2288:S
2289:C  2 END; { of set_serial }
2290:S
2291:S
2292:S
2293:S

```

```

2294:D  1 PROCEDURE clear_serial( select_code : type_isc ;
2295:D  2 line : type_serial_line);
2296:D  -4 VAR mybit : INTEGER;
2297:D  -8 VAR dummy : INTEGER;
2298:C  2 BEGIN
2299:C  2 WITH isc_table[select_code] DO BEGIN
2300:S
2301:C  3 IF card_type <> serial_card THEN io_escape(ioe_bad_sct,select_code);
2302:C  3
2303:C  3 IF isc_table[select_code].card_id = hp98628_async
2304:C  4 THEN BEGIN
2305:C  4 CASE line OF
2306:S
2307:C  5 rts_line: mybit := 1;
2308:S
2309:C  5 dtr_line: mybit := 2;
2310:C  5
2311:C  5 drs_line: mybit := 4;
2312:S
2313:C  5 OTHERWISE io_escape(ioe_bad_sct,select_code);
2314:S
2315:C  5 END; { of CASE line }
2316:C  4 dummy := iostatus(select_code,8);
2317:C  4 dummy := binand(dummy,bincmp(mybit));
2318:C  4 iocontrol(select_code,8*256,dummy);
2319:S
2320:C  4 END
2321:C  4 ELSE BEGIN
2322:C  4
2323:C  4 IF (card_id = hp98626) or (card_id = hp98644)
2324:C  5 THEN BEGIN
2325:C  5 CASE line OF
2326:S
2327:C  6 rts_line: mybit := 2;
2328:S
2329:C  6 dtr_line: mybit := 1;
2330:S
2331:C  6 drs_line: mybit := 8;
2332:S
2333:C  6 OTHERWISE io_escape(ioe_bad_sct,select_code);
2334:S
2335:C  6 END; { of CASE line }
2336:S
2337:C  5 dummy := iostatus(select_code,5);
2338:C  5 dummy := binand(dummy,bincmp(mybit));
2339:C  5 iocontrol(select_code,5,dummy);
2340:S
2341:C  5 END
2342:C  5 ELSE BEGIN
2343:C  5 CALL ( io_drv_ptr^.iod_clr ,
2344:C  5 io_imp_ptr ,
2345:C  5 ORD(line) );
2346:C  5 END; { of IF }
2347:S
2348:C  4 END; { of IF }
2349:S
2350:C  3 END; { of WITH isc_table BEGIN }
2351:C  2 END; { of clear_serial }
2352:S
2353:S

```



```

2354:S
2355:S
2356:S
2357:D      1  FUNCTION serial_line ( select_code : type_isc ;
2358:D      2      line : type_serial_line )
2359:D      3      : BOOLEAN ;
2360:D      4  VAR mybit : INTEGER;
2361:D      5  dummy : INTEGER;
2362:D      6  reg : INTEGER;
2363:D      7  mybool : BOOLEAN;
2364:C      8  BEGIN
2365:C      9  WITH isc_table[select_code] DO BEGIN
2366:C     10  IF card_type <> serial_card THEN io_escape(ioe_bad_sct,select_code);
2367:C     11  IF isc_table[select_code].card_id = hp98628_async
2368:C     12  THEN BEGIN
2369:C     13  CASE line OF
2370:C     14
2371:C     15      rts_line: BEGIN
2372:C     16          reg := 8;
2373:C     17          mybit := 0;
2374:C     18          END;
2375:C     19
2376:C     20      dtr_line: BEGIN
2377:C     21          reg := 8;
2378:C     22          mybit := 1;
2379:C     23          END;
2380:C     24
2381:C     25      drs_line: BEGIN
2382:C     26          reg := 8;
2383:C     27          mybit := 2;
2384:C     28          END;
2385:C     29
2386:C     30      dsr_line: BEGIN
2387:C     31          reg := 7;
2388:C     32          mybit := 0;
2389:C     33          END;
2390:C     34
2391:C     35      dcd_line: BEGIN
2392:C     36          reg := 7;
2393:C     37          mybit := 1;
2394:C     38          END;
2395:C     39
2396:C     40      cts_line: BEGIN
2397:C     41          reg := 7;
2398:C     42          mybit := 2;
2399:C     43          END;
2400:C     44
2401:C     45      ri_line: BEGIN
2402:C     46          reg := 7;
2403:C     47          mybit := 3;
2404:C     48          END;
2405:C     49
2406:C     50  OTHERWISE io_escape(ioe_bad_sct,select_code);
2407:C     51
2408:C     52  END; { of CASE line }
2409:C     53  dummy := iostatus(select_code,reg);
2410:C     54  mybool := bit_set(dummy,mybit);
2411:C     55
2412:C     56
2413:S

```

```

2414:C      4  END
2415:C      5  ELSE BEGIN
2416:C      6
2417:C      7  IF (card_id = hp98626) or (card_id = hp98644)
2418:C      8  THEN BEGIN
2419:C      9  CASE line OF
2420:C     10
2421:C     11      rts_line: BEGIN
2422:C     12          reg := 5;
2423:C     13          mybit := 1;
2424:C     14          END;
2425:C     15
2426:C     16      dtr_line: BEGIN
2427:C     17          reg := 5;
2428:C     18          mybit := 0;
2429:C     19          END;
2430:C     20
2431:C     21      drs_line: BEGIN
2432:C     22          reg := 5;
2433:C     23          mybit := 3;
2434:C     24          END;
2435:C     25
2436:C     26      dsr_line: BEGIN
2437:C     27          reg := 11;
2438:C     28          mybit := 5;
2439:C     29          END;
2440:C     30
2441:C     31      dcd_line: BEGIN
2442:C     32          reg := 11;
2443:C     33          mybit := 7;
2444:C     34          END;
2445:C     35
2446:C     36      cts_line: BEGIN
2447:C     37          reg := 11;
2448:C     38          mybit := 4;
2449:C     39          END;
2450:C     40
2451:C     41      ri_line: BEGIN
2452:C     42          reg := 11;
2453:C     43          mybit := 6;
2454:C     44          END;
2455:C     45
2456:C     46  OTHERWISE io_escape(ioe_bad_sct,select_code);
2457:C     47
2458:C     48  END; { of CASE line }
2459:C     49  dummy := iostatus(select_code,reg);
2460:C     50  mybool := bit_set(dummy,mybit);
2461:C     51
2462:C     52  END
2463:C     53  ELSE BEGIN
2464:C     54
2465:C     55  CALL ( io_drv_ptr^.iod_test ,
2466:C     56  io_tmp_ptr ,
2467:C     57  ORD(line) ,
2468:C     58  mybool );
2469:C     59
2470:C     60  END; { of IF }
2471:C     61
2472:C     62  END; { of IF }
2473:S

```



```

2474:C      3      END; ( of WITH isc_table BEGIN )
2475:C      2
2476:C      2      serial_line := mybool;
2477:S
2478:C      2      END; ( of serial_line )
2479:S
2480:S
2481:S
2482:S
2483:S
2484:C      1 END; ( of serial_0 )

```

```

2485:D      1 $PAGE$
2486:D      1 MODULE serial_3 ;
2487:S
2488:S      {
2489:S          date    07/22/81
2490:S          update  10/01/82
2491:S
2492:S          purpose This module contains the LEVEL
2493:S                  3 SERIAL GROUP procedures.
2494:S
2495:D      1      }
2496:S
2497:D      1 IMPORT   iodeclarations ;
2498:S
2499:D      1 EXPORT
2500:S
2501:S
2502:D      1  PROCEDURE set_baud_rate
2503:D      2      ( select_code : type_isc ;
2504:D      2        rate       : REAL );
2505:D      1  PROCEDURE set_stop_bits
2506:D      2      ( select_code : type_isc ;
2507:D      2        num_bits    : REAL );
2508:D      1  PROCEDURE set_char_length
2509:D      2      ( select_code : type_isc ;
2510:D      2        num_char_bit: INTEGER );
2511:D      1  PROCEDURE set_parity
2512:D      2      ( select_code : type_isc ;
2513:D      2        parity_mode : type_parity);
2514:D      1  PROCEDURE send_break
2515:D      2      ( select_code : type_isc );
2516:S
2517:D      1  PROCEDURE abort_serial
2518:D      2      ( select_code : type_isc );
2519:S
2520:S
2521:D      1 IMPLEMENT
2522:S
2523:S
2524:D      1  IMPORT   iocomasm ,
2525:D      1          general_0 ;
2526:S
2527:S
2528:D      -8 1  PROCEDURE set_baud_rate
2529:D      -8 2      ( select_code : type_isc ;
2530:D      -8 2        rate       : REAL );
2531:D      -12 2  VAR dummy : INTEGER;
2532:D      -8 2  FUNCTION calc_rate ( r : REAL ) : INTEGER;
2533:D      -12 3  VAR myrate : INTEGER;
2534:C      3  BEGIN
2535:C      3      myrate := 0;
2536:C      3      IF r=50 THEN myrate := 1;
2537:C      3      IF r=75 THEN myrate := 2;
2538:C      3      IF r=110 THEN myrate := 3;
2539:C      3      IF r=134.5 THEN myrate := 4;
2540:C      3      IF r=150 THEN myrate := 5;
2541:C      3      IF r=200 THEN myrate := 6;
2542:C      3      IF r=300 THEN myrate := 7;
2543:C      3      IF r=600 THEN myrate := 8;
2544:C      3      IF r=1200 THEN myrate := 9;

```



```

2545:C      3      IF r=1800 THEN myrate :=10;
2546:C      3      IF r=2400 THEN myrate :=11;
2547:C      3      IF r=3600 THEN myrate :=12;
2548:C      3      IF r=4800 THEN myrate :=13;
2549:C      3      IF r=9600 THEN myrate :=14;
2550:C      3      IF r=19200 THEN myrate :=15;
2551:S
2552:C      3      calc_rate := myrate;
2553:S
2554:C      3      END; { of calc_rate }
2555:C      2      BEGIN
2556:C      2      WITH isc_table[select_code] DO BEGIN
2557:S
2558:C      3      IF card_type <> serial_card THEN io_escape(ioe_misc,select_code);
2559:C      3
2560:C      3      IF isc_table[select_code].card_id = hp98628_async
2561:C      4      THEN BEGIN
2562:S
2563:C      4      dummy:=calc_rate(rate);
2564:C      4      IF dummy = 0 THEN io_escape(ioe_misc,select_code);
2565:C      4      iocontrol(select_code,20,dummy); { BUG 1270 TM 1/8/82 }
2566:C      4      iocontrol(select_code,21,dummy); { BUG 1270 TM 1/8/82 }
2567:S
2568:C      4      END
2569:C      4      ELSE BEGIN
2570:S
2571:C      4      IF (isc_table[select_code].card_id = hp98626) OR
2572:C      5      (isc_table[select_code].card_id = hp98644)
2573:C      5      THEN BEGIN
2574:S
2575:C      5      dummy:=ROUND(rate);
2576:C      5      IF dummy = 0 THEN io_escape(ioe_misc,select_code);
2577:C      5      iocontrol(select_code,3,dummy);
2578:S
2579:C      5      { what about 134.5 ? }
2580:S
2581:C      5      END
2582:C      5      ELSE BEGIN
2583:S
2584:C      5      io_escape(ioe_misc,select_code);
2585:S
2586:C      5      END; { of IF 98626 }
2587:S
2588:C      4      END; { of IF 98628_async }
2589:S
2590:C      3      END; { of WITH isc_table BEGIN }
2591:C      2      END; { of set_baud_rate }
2592:S
2593:S
2594:S
2595:S
2596:S
2597:D      -8      1      PROCEDURE set_stop_bits
2598:D      2      { select_code : type_isc ;
2599:D      2      { num_bits : REAL );
2600:D      -18      2      VAR myval : INTEGER;
2601:D      -16      2      dummy : INTEGER;
2602:C      2      BEGIN
2603:C      2      WITH isc_table[select_code] DO BEGIN
2604:S

```

```

2605:C      3      IF card_type <> serial_card THEN io_escape(ioe_misc,select_code);
2606:C      3
2607:C      3      IF isc_table[select_code].card_id = hp98628_async
2608:C      4      THEN BEGIN
2609:S
2610:C      4      IF num_bits = 1
2611:C      5      THEN BEGIN
2612:C      5      myval := 0;
2613:C      5      END
2614:C      5      ELSE BEGIN
2615:C      5      IF num_bits = 1.5
2616:C      6      THEN BEGIN
2617:C      6      myval := 1;
2618:C      6      END
2619:C      6      ELSE BEGIN
2620:C      6      IF num_bits = 2
2621:C      7      THEN BEGIN
2622:C      7      myval :=2
2623:C      7      END
2624:C      7      ELSE BEGIN
2625:C      7      io_escape(ioe_misc,select_code);
2626:C      7      END; { of IF 2 }
2627:S
2628:C      6      END; { of IF 1.5 }
2629:S
2630:C      5      END; { of IF 1 }
2631:S
2632:C      4      iocontrol(select_code,35,myval); { BUG 1270 TM 1/8/82 }
2633:S
2634:C      4      END
2635:C      4      ELSE BEGIN
2636:C      4      IF (isc_table[select_code].card_id = hp98626) { BUG 1269 TM 1/8/82 }
2637:C      5      OR (isc_table[select_code].card_id = hp98644)
2638:C      5      THEN BEGIN
2639:C      5      IF num_bits = 1
2640:C      6      THEN BEGIN
2641:C      6      myval:=0;
2642:C      6      END
2643:C      6      ELSE BEGIN
2644:C      6      IF num_bits = 1.5
2645:C      7      THEN BEGIN
2646:C      7      IF binand(iostatus(select_code,4),3)<>0
2647:C      8      THEN io_escape(ioe_misc,select_code);
2648:C      7      myval:=1;
2649:C      7      END
2650:C      7      ELSE BEGIN
2651:C      7      IF num_bits = 2
2652:C      8      THEN BEGIN
2653:C      8      myval:=1;
2654:C      8      END
2655:C      8      ELSE BEGIN
2656:C      8      io_escape(ioe_misc,select_code);
2657:C      8      END; { of IF 2 }
2658:S
2659:C      7      END; { of IF 1.5 }
2660:S
2661:C      6      END; { of IF 1 }
2662:S
2663:C      5      dummy:=iostatus(select_code,4);
2664:C      5      dummy:=binand(dummy,255)+myval*4; { 0359 TM 8/26/82 }

```



```

2665:C      5      iocontrol(select_code,4,dummy);
2666:S
2667:C      5      END
2668:C      5      ELSE BEGIN
2669:C      5      io_escape(ioe_misc,select_code);
2670:C      5      END; { of IF 98626 }
2671:S
2672:C      4      END; { of IF 98628_async }
2673:S
2674:C      3      END; { of WITH isc_table BEGIN }
2675:S
2676:C      2      END; { set_stop_bits }
2677:S
2678:S
2679:S
2680:S
2681:D      1      PROCEDURE set_char_length
2682:D      2      ( select_code : type_isc ;
2683:D      2      num_char_bit: INTEGER );
2684:D      -4      VAR myval : INTEGER;
2685:D      -8      dummy : INTEGER;
2686:C      2      BEGIN
2687:C      2      WITH isc_table[select_code] DO BEGIN
2688:S
2689:C      3      IF card_type <> serial_card THEN io_escape(ioe_misc,select_code);
2690:S
2691:C      3      CASE num_char_bit OF
2692:S
2693:C      4      5: myval := 0;
2694:C      4      6: myval := 1;
2695:C      4      7: myval := 2;
2696:C      4      8: myval := 3;
2697:S
2698:C      4      OTHERWISE io_escape(ioe_misc,select_code);
2699:S
2700:C      4      END; { of CASE }
2701:S
2702:S
2703:C      3      IF isc_table[select_code].card_id = hp98628_async
2704:C      4      THEN BEGIN
2705:C      4      iocontrol(select_code,34,myval);
2706:C      4      END
2707:C      4      ELSE BEGIN
2708:C      4      IF(isc_table[select_code].card_id = hp98626)
2709:C      5      or (isc_table[select_code].card_id = hp98644)
2710:C      5      THEN BEGIN
2711:C      5      dummy:=iostatus(select_code,4);
2712:C      5      dummy:=binand(dummy,252)+myval;
2713:C      5      iocontrol(select_code,4,dummy);
2714:C      5      { 0359 TM 8/23/82 }
2715:C      5      { 557 TM 10/1/82 }
2716:C      5      END
2717:C      5      ELSE BEGIN
2718:C      5      io_escape(ioe_misc,select_code);
2719:C      5      END
2720:C      5      END; { of IF 98626 }
2721:S
2722:S
2723:S
2724:C      5

```

```

2725:S
2726:C      4      END; { of IF 98628_async }
2727:S
2728:C      3      END; { of WITH isc_table BEGIN }
2729:C      2      END; { set_char_length }
2730:S
2731:S
2732:S
2733:S
2734:S
2735:D      1      PROCEDURE set_parity
2736:D      2      ( select_code : type_isc ;
2737:D      2      parity_mode : type_parity);
2738:D      -4      VAR myval : INTEGER;
2739:D      -8      dummy : INTEGER;
2740:C      2      BEGIN
2741:C      2      WITH isc_table[select_code] DO BEGIN
2742:S
2743:C      3      IF card_type <> serial_card THEN io_escape(ioe_misc,select_code);
2744:C      3      IF isc_table[select_code].card_id = hp98628_async
2745:C      4      THEN BEGIN
2746:C      4      CASE parity_mode OF
2747:S
2748:C      5      no_parity: myval := 0;
2749:C      5      odd_parity: myval := 1;
2750:C      5      even_parity: myval := 2;
2751:C      5      zero_parity: myval := 3;
2752:C      5      one_parity: myval := 4;
2753:C      5      { 0355 TM 8/20/82 }
2754:C      5      { 0355 TM 8/20/82 }
2755:S
2756:C      5      OTHERWISE io_escape(ioe_misc,select_code);
2757:S
2758:C      5      END; { of CASE }
2759:S
2760:C      4      iocontrol(select_code,36,myval);
2761:S
2762:C      4      END
2763:C      4      ELSE BEGIN
2764:C      4      IF (isc_table[select_code].card_id = hp98626)
2765:C      5      or (isc_table[select_code].card_id = hp98644)
2766:C      5      THEN BEGIN
2767:C      5      CASE parity_mode OF
2768:S
2769:C      6      no_parity: myval := 0;
2770:C      6      odd_parity: myval := 1;
2771:C      6      even_parity: myval := 3;
2772:C      6      one_parity: myval := 5;
2773:C      6      zero_parity: myval := 7;
2774:C      6      OTHERWISE io_escape(ioe_misc,select_code);
2775:C      6      END; { of CASE }
2776:C      6      dummy:=iostatus(select_code,4);
2777:C      5      dummy:=binand(dummy,195)+myval+8;
2778:C      5      iocontrol(select_code,4,dummy);
2779:C      5      { 0359 TM 8/23/82 }
2780:S
2781:S
2782:S
2783:S
2784:S

```



```

2785:C      5      END
2786:C      5      ELSE BEGIN
2787:S
2788:C      5      io_escape(ioe_misc,select_code);
2789:S
2790:C      5      END; { of IF 98626 }
2791:S
2792:C      4      END; { of IF 98628_async }
2793:S
2794:C      3      END; { of WITH isc_table BEGIN }
2795:C      2      END; { set_parity }
2796:S
2797:S
2798:S
2799:S
2800:C
2801:C      1  PROCEDURE send_break
2802:C      2      ( select_code : type_isc );
2803:C      2      BEGIN
2804:C      2      { what about active tfrs }
2805:S
2806:C      2      WITH isc_table[select_code] DO BEGIN
2807:S
2808:C      3      IF card_type <> serial_card THEN io_escape(ioe_misc,select_code);
2809:C      3
2810:C      3      IF isc_table[select_code].card_id = hp98628_async
2811:C      4      THEN BEGIN
2812:S
2813:C      4      iocontrol(select_code,6,1);
2814:S
2815:C      4      END
2816:C      4      ELSE BEGIN
2817:C      4      IF (card_id = hp98626) or (card_id = hp98644)
2818:C      5      THEN BEGIN
2819:S
2820:C      5      iocontrol(select_code,1,1);
2821:S
2822:C      5      END
2823:C      5      ELSE BEGIN
2824:C      5      io_escape(ioe_misc,select_code);
2825:C      5
2826:C      5      END; { of IF }
2827:C      5
2828:S
2829:C      4      END; { of IF }
2830:S
2831:C      3      END; { of WITH isc_table BEGIN }
2832:C      2      END; { of send_break }
2833:S
2834:S
2835:C      1  PROCEDURE abort_serial
2836:C      2      ( select_code : type_isc );
2837:C      2      BEGIN
2838:S
2839:C      2      { what about active tfrs }
2840:S
2841:C      2      WITH isc_table[select_code] DO BEGIN
2842:S
2843:C      3      IF card_type <> serial_card THEN io_escape(ioe_misc,select_code);
2844:C      3

```

```

2845:C      3      IF isc_table[select_code].card_id = hp98628_async
2846:C      4      THEN BEGIN
2847:S
2848:C      4      iocontrol(select_code,256*125,1);          { BUG xxxx TM 1/26/82 }
2849:S
2850:C      4      END
2851:C      4      ELSE BEGIN
2852:C      4      IF (card_id = hp98626) or (card_id = hp98644) { BUG FIX 6/4/84 }
2853:C      5      THEN BEGIN
2854:S
2855:C      5      iocontrol(select_code,0,1);
2856:S
2857:C      5      END
2858:C      5      ELSE BEGIN
2859:S
2860:C      5      io_escape(ioe_misc,select_code);
2861:C      5
2862:C      5      END; { of IF }
2863:S
2864:C      4      END; { of IF }
2865:S
2866:C      3      END; { of WITH isc_table BEGIN }
2867:C      2      END; { of abort_serial }
2868:S
2869:S
2870:S
2871:C      1  END.      { of serial_3 }
2872:S

```

No errors. 1 warnings.

***** Nonstandard language features enabled *****

KERNEL

Description

This is the I/O library kernel.

Usage

The modules in KERNEL provide the framework for the rest of the I/O library and I/O drivers.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```
1:0      0
2:0      (*
3:0
4:0      (c) Copyright Hewlett-Packard Company, 1984.
5:0      All rights are reserved. Copying or other
6:0      reproduction of this program except for archival
7:0      purposes is prohibited without the prior
8:0      written consent of Hewlett-Packard Company.
9:0
10:0
11:0
12:0
13:0      RESTRICTED RIGHTS LEGEND
14:0      Use, duplication, or disclosure by the Government
15:0      is subject to restrictions as set forth in
16:0      paragraph (b) (3) (B) of the Rights in Technical
17:0      Data and Computer Software clause in
18:0      DAR 7-104.9(a).
19:0
20:0      HEWLETT-PACKARD COMPANY
21:0      0 Fort Collins, Colorado      *)
22:0
23:0      0 $MODCAL ON$
24:0      0 $PARTIAL EVAL ON$
25:0      0 $RANGE OFF$
26:0      0 $DEBUG OFF$
27:0      0 $OVFLCHECK OFF$
28:0      0 $STACKCHECK OFF$
```

```
29:0      0 $PAGES$
30:0      0 {*****}
31:0      0 {
32:0      0 {      RELEASED      VERSION      3.0      }
33:0      0 {
34:0      0 {*****}
35:0      0 {
36:0      0 {
37:0      0 {      IOLIB      KERNEL      }
38:0      0 {
39:0      0 {*****}
40:0      0 {
41:0      0 {
42:0      0 {
43:0      0 {      library      - IOLIB      }
44:0      0 {      name      - KERNEL      }
45:0      0 {      module(s)      - iodeclarations      }
46:0      0 {      - iocomasm      }
47:0      0 {      - general_0      }
48:0      0 {
49:0      0 {      author      -
50:0      0 {      phone      -
51:0      0 {
52:0      0 {      date      - June 1 , 1981      }
53:0      0 {      update      - Feb 9 , 1984      }
54:0      0 {      release      - xxxxxxxxxxxxxxxx      (3.0 release date)
55:0      0 {
56:0      0 {      source      - IOLIB:KERNEL.TEXT      }
57:0      0 {      object      - IOLIB:KERNEL.CODE      }
58:0      0 {
59:0      0 {*****}
```

```
60:0      0 $PAGES$
61:0      0 {*****}
62:0      0 {
63:0      0 {
64:0      0 {      This is the source code for an external procedures library
65:0      0 {      to be used for general purpose interfacing on the HP 9826.
66:0      0 {
67:0      0 {      The library consists of 3 primary sets of modules -
68:0      0 {
69:0      0 {          1.      KERNEL modules
70:0      0 {          2.      driver modules
71:0      0 {          3.      IOLIB modules
72:0      0 {
73:0      0 {      The KERNEL modules consist of the following modules -
74:0      0 {
75:0      0 {          1.      iodeclarations      ( contains static r/w space )
76:0      0 {          2.      iocomasm
77:0      0 {          3.      general_0      ( initialization & low level
78:0      0 {          routines like ioread/iowrite )
79:0      0 {
80:0      0 {      The KERNEL modules also have an executable program segment
81:0      0 {      that gets executed at the time it is loaded. This program
82:0      0 {      initializes the static read/write memory. This program also
83:0      0 {      allocates the temporary storage for any card that exists -
84:0      0 {      independent of whether there is or is not a driver for it.
85:0      0 {
86:0      0 {      The driver modules consist of the actual assembly or PASCAL
87:0      0 {      routines that deal with a specific interface card. There is
88:0      0 {      also an executable program segment for each driver module.
89:0      0 {
```



```

89:D 0 (* initialized by the KERNEL general_0 module for all select codes *)
90:D 0 (* that have the right interface card ( HP1B drivers will search *)
91:D 0 (* for the 98624 interface ). This program will then set up the *)
92:D 0 (* driver tables to point to the correct drivers. *)
93:D 0 (*
94:D 0 (* The rest of the IDLI8 modules are high-level modules that are *)
95:D 0 (* used by an end user in his/her application program. *)
96:D 0 (*
97:D 0 (* The KERNEL and some set of driver modules will exist in the *)
98:D 0 (* INITLI8 file as object code ( not EXPORT text ). The *)
99:D 0 (* export text will reside on the IO file. The rest *)
100:D 0 (* of the library will reside on the IO file. *)
101:D 0 (*
102:D 0 (*****

```

```

103:D 0 SPAGES
104:D 0 *****
105:D 0
106:D 0
107:D 0      BUG FIX HISTORY      - after release 1.0
108:D 0
109:D 0
110:D 0      BUG #      8Y / ON      LOC      DESCRIPTION
111:D 0      -----
112:D 0
113:D 0      1281      -----      IODECLARATIONS      bad error message.
114:D 0      01/08/82      No code change in this
115:D 0      module - code change in
116:D 0      GENERAL_3.
117:D 0
118:D 0      aaaa      -----      kernel
119:D 0      05/24/82      initialize      The CRT and keyboard
120:D 0      select codes were
121:D 0      interchanged. Should
122:D 0      have no effect.
123:D 0
124:D 0      bbbb      -----      kernel
125:D 0      07/09/82      initialize      No bug sheet.
126:D 0      iodeclarations      Adding machine id info
127:D 0      and the
128:D 0      Ganglia drivers.
129:D 0
130:D 0      jw      -----      kernel
131:D 0      07/12/82      initialize      No bug sheet.
132:D 0      iodeclarations      Adding temp type for
133:D 0      626 drivers.
134:D 0
135:D 0      0082      -----      kernel
136:D 0      07/23/82      initialize      Adding a link for the
137:D 0      iodeclarations      error message function.
138:D 0      end_error_link      See also GENERAL_3.
139:D 0
140:D 0      cccc      -----      iodeclarations      Allowing the 98629 card
141:D 0      08/16/82      and the 98628 dsnd1 as
142:D 0      identifiable cards.
143:D 0      See DC_DRV modules also.
144:D 0
145:D 0      0350      -----      general_0      Fundamental flaw with
146:D 0      08/19/82      the dummy drivers.
147:D 0
148:D 0      0364      -----      iodeclarations      Addition of SRM errors.
149:D 0      08/23/82      See also GENERAL_3.
150:D 0
151:D 0      0367      -----      iolibrary_kernel
152:D 0      09/22/82      Allow kernel to install
153:D 0      itself - probably will
154:D 0      not be used much.
155:D 0
156:D 0      tttt      -----      kbd_rdb      Change in the handling
157:D 0      09/22/82      of the keyboard end of
158:D 0      line condition from 1.0.
159:D 0
160:D 0      uuuu      -----      iodeclarations      Differentiate between a
161:D 0      09/28/82      628 async and 629 srm
162:D 0      card type (srm/serial).
163:D 0      See also DC_DRV.

```



```
163:D 0 (* jws ----- iodeclarations Add card id's and types *)
164:D 0 (* 03/25/83 general_0 for EPROM programmer *)
165:D 0 (* and bubble cards. *)
166:D 0 (* Add initialization code. *)
167:D 0 (*
168:D 0 (* jws2 ----- general_0 Add test for internal *)
169:D 0 (* 06/28/83 HP-18 present. *)
170:D 0 (*
171:D 0 (* jws3 ----- iodeclarations Deleted io_model_number *)
172:D 0 (* 02/09/84 general_0 and io_model_name stuff *)
173:D 0 (* Added GATOR id, init. *)
174:D 0 (* jws4 ----- iodeclarations Added id and init for *)
175:D 0 (* 03/05/84 general_0 9864 card *)
176:D 0 (*****
177:S
```

```
178:D 0 SPAGES
179:D 0 (*****
180:D 0 (*
181:D 0 (*
182:D 0 (* REFERENCES :
183:D 0 (*
184:D 0 (*
185:D 0 (* 1. 9826 I/O Designers Guide ( ----- )
186:D 0 (*
187:D 0 (* 2. 68000 Manual ( Motorola )
188:D 0 (*
189:D 0 (* 3. Pascal alpha site ERS ( ----- )
190:D 0 (*
191:D 0 (* 4. Pascal I/O Library ERS ( ----- )
192:D 0 (*
193:D 0 (* 5. 9826 HPL EIO & IOD listings ( ----- )
194:D 0 (*
195:D 0 (* 6. 9826 HPL Misc. I/O Doc. ( ----- )
196:D 0 (*
197:D 0 (* 7. 9826 card documentation ( ----- )
198:D 0 (*
199:D 0 (*
200:D 0 (*****
```



```

201:D      0 $PAGES
202:D      0 PROGRAM iolibary_kernel ( INPUT , OUTPUT );
203:S
204:S
205:S
206:S      { the PROGRAM surrounds the following modules because there needs
207:S      to be a start address for this set of modules to allow
208:D      1      initialization to occur }
209:S

```

```

210:D      1 $PAGES
211:D      1 (*****
212:D      1 (*
213:D      1 (*
214:D      1 (*
215:D      1 (*
216:D      1 (*
217:D      1 (*****
218:S
219:S
220:S
221:S
222:S      1 MODULE iodeclarations ;
223:S
224:S      {
225:S      date 07/15/81
226:S      update 03/05/84
227:S
228:S      purpose This module contains the common
229:S      declarations to be used by the
230:S      rest of the I/O library.
231:S
232:D      1      }
233:S
234:S
235:S
236:D      1 IMPORT sysglobals;
237:S
238:D      1 EXPORT
239:D      1
240:D      1 CONST
241:D      1      iominisc = 0;          { 0 - 7 internal }
242:D      1      iomaxisc = 31;        { 8 - 31 external 8..31 }
243:D      1      minrealisc = 7;
244:D      1      maxrealisc = 31;      { 7..31 are real isc.s with temps }
245:S
246:D      1      io_line_feed = CHR(10);
247:D      1      io_carriage_rtn = CHR(13);
248:S

```



```

249:D      1      $PAGES
250:S
251:D      1      { escape code constants }
252:S
253:D      1      ioescapecode   = -26;
254:S
255:D      1      ioe_no_error    = 0;      { no error }
256:D      1      ioe_no_card     = 1;      { no card at select code }
257:D      1      ioe_not_hpib    = 2;      { interface should be hpib }
258:D      1      ioe_not_act     = 3;      { not active controller }
259:D      1      ioe_not_dvc     = 4;      { should be device not isc }
260:D      1      ioe_no_space   = 5;      { no space left in buffer }
261:D      1      ioe_no_data    = 6;      { no data left in buffer }
262:D      1      ioe_bad_tfr    = 7;      { improper transfer attempted }
263:D      1      ioe_isc_busy    = 8;      { the select code is busy }
264:D      1      ioe_buf_busy    = 9;      { the buffer is busy }
265:D      1      ioe_bad_cnt     = 10;     { improper transfer count }
266:D      1      ioe_bad_tmo    = 11;     { bad timeout value }
267:D      1      ioe_no_driver   = 12;     { no driver for this card }
268:D      1      ioe_no_dma     = 13;     { no dma }
269:D      1      ioe_no_word    = 14;     { word operations not allowed }
270:D      1      ioe_not_talk   = 15;     { not addressed as talker }
271:D      1      ioe_not_lsn    = 16;     { not addressed as listener }
272:D      1      ioe_timeout    = 17;     { a timeout has occurred }
273:D      1      ioe_not_sctl   = 18;     { not system controller }
274:D      1      ioe_rds_wtc    = 19;     { bad status or control }
275:D      1      ioe_bad_sctl   = 20;     { bad sel/clear/test operation }
276:D      1      ioe_crg_dwn    = 21;     { interface card is dead }
277:D      1      ioe_eod_seen   = 22;     { end/eod has occurred }
278:D      1      ioe_misc      = 23;     { miscellaneous - }
279:D      1      { value of param error }
280:S
281:D      1      ioe_sr_toomany  = 304;    { too many chars w/o terminator } { 0364 TM 8/23/82 }
282:D      1      ioe_dc_fail    = 306;    { dc interface failure }
283:D      1      ioe_dc_usart    = 313;    { USART receive buffer overflow }
284:D      1      ioe_dc_ovfl    = 314;    { receive buffer overflow }
285:D      1      ioe_dc_clk     = 315;    { missing clock }
286:D      1      ioe_dc_cts     = 316;    { CTS false too long }
287:D      1      ioe_dc_car     = 317;    { lost carrier disconnect }
288:D      1      ioe_dc_act     = 318;    { no activity disconnect }
289:D      1      ioe_dc_conn    = 319;    { connection not established }
290:D      1      ioe_dc_conf    = 325;    { bad data bits/par combination }
291:D      1      ioe_dc_reg     = 326;    { bad status/control register }
292:D      1      ioe_dc_rval    = 327;    { control value out of range }
293:D      1      ioe_sr_fail    = 353;    { data link failure } { 0364 TM 8/23/82 }
294:S
295:D      1      no_isc         = 255;    { used for ioe_isc within io errors }
296:S

```

```

297:D      1      $PAGES
298:D      1      { hpib message constants }
299:S
300:D      1      gtl_message     = CHR( 1 );
301:D      1      sdc_message     = CHR( 4 );
302:D      1      ppc_message     = CHR( 5 );
303:D      1      get_message     = CHR( 8 );
304:D      1      tct_message     = CHR( 9 );
305:D      1      llo_message     = CHR( 17 );
306:D      1      dcl_message     = CHR( 20 );
307:D      1      ppu_message     = CHR( 21 );
308:D      1      spe_message     = CHR( 24 );
309:D      1      spd_message     = CHR( 25 );
310:D      1      unl_message     = CHR( 63 );
311:D      1      unt_message     = CHR( 95 );
312:D      1      ppe_message     = CHR( 96 );
313:D      1      ppd_message     = CHR(112 );
314:S
315:S
316:D      1      talk_constant   = 64;
317:D      1      listen_constant = 32;
318:S

```



```

319:D 1 $PAGE$
320:S
321:D 1 { card type constants }
322:S
323:D 1 no_card = 0 ;
324:D 1 other_card = 1 ;
325:D 1 system_card = 2 ;
326:D 1 hpib_card = 3 ;
327:D 1 gpio_card = 4 ;
328:D 1 serial_card = 5 ;
329:D 1 graphics_card = 6 ;
330:D 1 srm_card = 7 ;
331:D 1 bubble_card = 8 ; { shared resource mgr } { uuuu TM 9/28/82 }
332:D 1 eprom_prgmr = 9 ; { bubble memory } { jws 3/25/83 }
333:S { eprom programmer } { jws 3/25/83 }
334:S
335:D 1 { card id constants }
336:D 1 { positive id's are the actual }
337:D 1 { card id's }
338:S
339:D 1
340:D 1 hp98628_dsnd1 = -7 ; { DSN/DL }
341:D 1 hp98629 = -6 ; { shared resource mgr } { cccc TM 8/16/82 }
342:D 1 hp_datacomm = -5 ; { cccc TM 8/16/82 }
343:D 1 hp98620 = -4 ;
344:D 1 internal_kbd = -3 ;
345:D 1 internal_crt = -2 ;
346:D 1 internal_hpib = -1 ;
347:S
348:D 1 no_id = 0 ;
349:S
350:D 1 hp98624 = 1 ; { hpib }
351:D 1 hp98626 = 2 ; { serial }
352:D 1 hp98622 = 3 ; { gpio }
353:D 1 hp98623 = 4 ; { bcd }
354:D 1 hp98625 = 8 ; { disk }
355:D 1 hp98628_async = 20 ; { hp98628 }
356:D 1 hpGATOR = 25 ; { bitmap display -- need number jws3 }
357:D 1 hp98253 = 27 ; { EPROM programmer } { jws 3/25/83 }
358:D 1 hp98627 = 28 ; { graphics }
359:D 1 hp98259 = 30 ; { bubble memory } { jws 3/25/83 }
360:D 1 hp98644 = 66 ; { serial -- pri. id=2, sec. id=2 jws4 }
361:S

```

```

362:D 1 $PAGE$
363:S
364:D 1 TYPE
365:S
366:D 1 { general declarations }
367:S
368:D 1 type_isc = iominisc..iomaxisc ;
369:D 1 type_device = iominisc..iomaxisc*100+99;
370:D 1 io_bit = 0..15 ;
371:D 1 io_byte = 0..255 ;
372:D 1 io_word = -32768..32767 ;
373:D 1 io_string = STRING(255);
374:D 1 io_proc_type = PROCEDURE;
375:S
376:D 1 errlink_type = PROCEDURE ( errorcode : INTEGER ; { 0082 TM 7/23/82 }
377:D 1 VAR s : io_string ); { 0082 TM 7/23/82 }
378:S
379:D 1 { card id and type declarations }
380:S
381:D 1 type_of_card = io_word;
382:D 1 type_card_id = io_word;
383:S
384:D 1 { hpib type declarations }
385:S
386:D 1 type_hpib_addr = 0..31 ;
387:D 1 type_hpib_line = ( ren_line ,
388:D 1 ifc_line ,
389:D 1 srq_line ,
390:D 1 eol_line ,
391:D 1 nrfd_line ,
392:D 1 ndac_line ,
393:D 1 dav_line ,
394:D 1 atn_line ,
395:D 1 );
396:D 1
397:S
398:D 1 { serial type declarations }
399:S
400:D 1 type_parity = ( no_parity ,
401:D 1 odd_parity ,
402:D 1 even_parity ,
403:D 1 zero_parity ,
404:D 1 one_parity ,
405:D 1 );
406:D 1
407:D 1 type_serial_line= ( rts_line ,
408:D 1 cts_line ,
409:D 1 dcd_line ,
410:D 1 dsr_line ,
411:D 1 dtr_line ,
412:D 1 ri_line ,
413:D 1 dir_line ,
414:D 1 );

```



```

415:D 1 $PAGE$
416:D 1 { driver declarations }
417:D
418:D 1 io_proc = PROCEDURE ( temp : ANYPTR );
419:D 1 io_proc_c = PROCEDURE ( temp : ANYPTR;
420:D 1 v : CHAR );
421:D 1 io_proc_vc = PROCEDURE ( temp : ANYPTR;
422:D 1 VAR v : CHAR );
423:D 1 io_proc_w = PROCEDURE ( temp : ANYPTR;
424:D 1 v : io_word );
425:D 1 io_proc_vw = PROCEDURE ( temp : ANYPTR;
426:D 1 VAR v : io_word );
427:D 1 io_proc_s = PROCEDURE ( temp : ANYPTR;
428:D 1 reg : io_word;
429:D 1 v : io_word );
430:D 1 io_proc_vs = PROCEDURE ( temp : ANYPTR;
431:D 1 reg : io_word;
432:D 1 VAR v : io_word );
433:D 1 io_proc_l = PROCEDURE ( temp : ANYPTR;
434:D 1 line : io_bit );
435:D 1 io_proc_vl = PROCEDURE ( temp : ANYPTR;
436:D 1 line : io_bit;
437:D 1 VAR v : BOOLEAN );
438:D 1 io_proc_vb = PROCEDURE ( temp : ANYPTR;
439:D 1 VAR v : BOOLEAN );
440:D 1 io_proc_ptr = PROCEDURE ( temp : ANYPTR;
441:D 1 v : ANYPTR );
442:D
443:D 1 drv_table_type = RECORD
444:D 1 iod_init : io_proc;
445:D 1 iod_isr : ISRPROCTYPE;
446:D 1 iod_rdb : io_proc_vc;
447:D 1 iod_wtb : io_proc_c;
448:D 1 iod_rdw : io_proc_vw;
449:D 1 iod_wtw : io_proc_w;
450:D 1 iod_rds : io_proc_vs;
451:D 1 iod_wtc : io_proc_s;
452:D 1 iod_end : io_proc_vb;
453:D 1 iod_tfr : io_proc_ptr;
454:D 1 iod_send : io_proc_c;
455:D 1 iod_ppoll : io_proc_vc;
456:D 1 iod_set : io_proc_l;
457:D 1 iod_clr : io_proc_l;
458:D 1 iod_test : io_proc_vl;
459:D 1
460:D
461:D
462:D 1 { procedure definition for DMA termination procedure }
463:D 1 io_funny_proc = RECORD
464:D 1 CASE BOOLEAN OF
465:D 1 TRUE :
466:D 1 ( real_proc : io_proc );
467:D 1 FALSE :
468:D 1 ( dummy_pr : ANYPTR;
469:D 1 dummy_sl : ANYPTR );
470:D 1
471:D 1 END;
472:D 1 { procedure definition for user EOT/ISR procedures }
473:D
474:D 1 { Note that the current definition of EOT / ISR procedures is

```

```

475:S that they will execute ( probably ) inside an ISR and will
476:S therefore be at that interrupt level. This has several side
477:S effects - such as READ from the keyboard not working- since they
478:S depend on interrupts occurring.
479:S
480:S The ISR / EOT procedures will work best if they set a flag that
481:S the user program will periodically check. Basically this means
482:S the user needs to implement his own end-of-line searching.
483:S
484:S The ISR / EOT procedure is allowed to have a NON-ZERO static link.
485:S This means it can be a nested procedure. This is potentially
486:S dangerous if the user program is no longer in that chain.
487:D 1 CAVERT EHPTOR.
488:S
489:S
490:D 1 { interface driver space }
491:D
492:D 1 io_temp_type = PACKED RECORD
493:D 1 myisrib : ISIRB;
494:D 1 user_isr : io_funny_proc; {JPC 2/22/82}
495:D 1 user_parm : ANYPTR; {JPC 2/22/82}
496:D 1 card_addr : ANYPTR;
497:D 1 in_bufptr : ANYPTR;
498:D 1 out_bufptr : ANYPTR;
499:D 1 eirByte : CHAR;
500:D 1 my_isc : io_byte;
501:D 1 timeout : INTEGER; { in milliseconds }
502:D 1 addressed : io_word;
503:D 1 drv_misc : ARRAY[1..32] OF CHAR;
504:D 1
505:D 1 END;
506:D
507:D 1 io_temp_type2 = PACKED RECORD
508:D 1 myisrib : ISIRB;
509:D 1 user_isr : io_funny_proc; {JPC 2/22/82}
510:D 1 user_parm : ANYPTR; {JPC 2/22/82}
511:D 1 card_addr : ANYPTR;
512:D 1 in_bufptr : ANYPTR;
513:D 1 out_bufptr : ANYPTR;
514:D 1 eirByte : CHAR;
515:D 1 my_isc : io_byte;
516:D 1 timeout : INTEGER; { in milliseconds }
517:D 1 addressed : io_word;
518:D 1 drv_misc : ARRAY[1..128] OF CHAR;
519:D 1
520:D 1 END;
521:D
522:D 1 io_temp_type3 = PACKED RECORD {Jw 7/12/82}
523:D 1 myisrib : ISIRB; {Jw 7/12/82}
524:D 1 user_isr : io_funny_proc; {JPC 2/22/82}
525:D 1 user_parm : ANYPTR; {JPC 2/22/82}
526:D 1 card_addr : ANYPTR; {Jw 7/12/82}
527:D 1 in_bufptr : ANYPTR; {Jw 7/12/82}
528:D 1 out_bufptr : ANYPTR; {Jw 7/12/82}
529:D 1 eirByte : CHAR; {Jw 7/12/82}
530:D 1 my_isc : io_byte; {Jw 7/12/82}
531:D 1 timeout : INTEGER; { in milliseconds }
532:D 1 addressed : io_word; {Jw 7/12/82}
533:D 1 drv_misc : ARRAY[1..160] OF CHAR; {Jw 7/12/82}
534:D 1

```



```

S35:D      1      pio_tmp_ptr      = ^io_temp_type;
S36:S
S37:S
S38:D      1
S39:D      1      isc_table_type = RECORD
S40:D      1          io_drv_ptr: ^drv_table_type;
S41:D      1          io_tmp_ptr: pio_tmp_ptr;
S42:D      1          card_type : type_of_card;
S43:D      1          user_time : INTEGER;
S44:D      1          card_id   : type_card_id;
S45:D      1          card_ptr  : ANYPTR;
                END;

```

```

S46:D      1      $PAGES
S47:D      1      { transfer declarations }
S48:S
S49:D      1      user_tfr_type = ( dummy_tfr_1 ,          { serial INTR }
S50:D      1          serial_DMA ,
S51:D      1          serial_FHS ,
S52:D      1          serial_FASTEST ,
S53:D      1          dummy_tfr_2 ,          { serial OVERLAP }
S54:D      1          overlap_INTR ,
S55:D      1          overlap_DMA ,
S56:D      1          overlap_FHS ,
S57:D      1          overlap_FASTEST ,
S58:D      1          OVERLAP );
S59:D      1      actual_tfr_type = ( no_tfr ,
S60:D      1          INTR_tfr ,
S61:D      1          DMA_tfr ,
S62:D      1          BURST_tfr ,
S63:D      1          FHS_tfr );
S64:D      1      dir_of_tfr = ( to_memory ,          { input = BOOLEAN false }
S65:D      1          from_memory ,          { output = BOOLEAN true }
S66:D      1          );
S67:S
S68:D      1      buf_type = PACKED ARRAY[0..maxint] OF CHAR;
S69:S
S70:D      1      buf_info_type = RECORD
S71:D      1          drv_tmp_ptr : pio_tmp_ptr;
S72:D      1          active_isc : io_byte;
S73:D      1          act_tfr : actual_tfr_type ;
S74:D      1          usr_tfr : user_tfr_type ;
S75:D      1          b_w_mode : BOOLEAN ;          { word = BOOLEAN true }
S76:D      1          end_mode : BOOLEAN ;          { eoi = BOOLEAN true }
S77:D      1          direction : dir_of_tfr ;
S78:D      1          term_char : -1..255 ;          { -1 = no termination char }
S79:D      1          term_count : INTEGER ;
S80:D      1          buf_ptr : ^buf_type ;
S81:D      1          buf_size : INTEGER ;
S82:D      1          buf_empty : ANYPTR ;
S83:D      1          buf_fill : ANYPTR ;
S84:D      1          eot_proc : io_funny_proc;      {JPC 2/22/82}
S85:D      1          eot_parm : ANYPTR ;          {JPC 2/22/82}
S86:D      1          dma_priority: 800LEAN;
S87:D      1      END;

```



```

588:D      1  $PAGE$
589:D      1  VAR
590:S
591:S      { dma driver variables - used by DMA_DRV module
592:S      and by assembly language drivers.
593:D      1  ----- DON'T MOVE ----- )
594:S
595:D      -8  1      dma_ch_0      : io_funny_proc ;
596:D      -10 1      dma_isc_0      : io_byte ;
597:D      -18 1      dma_ch_1      : io_funny_proc ;
598:D      -20 1      dma_isc_1      : io_byte ;
599:D      -40 1      dma_isrIb0      : ISRIB ;
600:D      -60 1      dma_isrIb1      : ISRIB ;
601:D      -61 1      dma_here      : BODLEARN;
602:S
603:S
604:S
605:D      -62 1      io_work_char    : CHAR;
606:S
607:S
608:D      -82 1      { io escape access variables }
609:S
610:D      -66 1      ioe_result      : INTEGER;
611:S
612:S      ioe_isc      : INTEGER;      { must be integer because the sc could
613:D      -70 1                                     be no sc ( 255 ) or a device
614:S                                         like 701 etc. }
615:S
616:D      -70 1      isc_table      : PACKED ARRAY [type_isc]
617:D      -710 1      OF isc_table_type;
618:S
619:S      io_revid      : STRING[96]; { revision string - added 2/5/82 - TM
620:S
621:S      meaning - 'IO 1.0 refers to the IO
622:S      library ( in system.library ) and
623:S      the kernel code. Each driver
624:S      module will append an indication of
625:S      its revision like 'G1.0' for GP10.
626:S      So - a typical system would have a
627:S      io_revid of 'IO 1.0 : D1.0 H1.0
628:S      G1.0 S1.0'.
629:S
630:S      known ids -
631:S
632:S      main io lib 'IO 1.0 : '
633:S      dma driver  ' D1.0'
634:S      hpib driver  ' H1.0'
635:S      gpib driver  ' G1.0'
636:S      828 driver   ' S1.0'
637:S      626 driver   ' R1.0'
638:D      -808 1      )
639:S
640:S
641:D      -816 1      io_error_link    : errlnk_type; { error msg extension } { 0082 TM 7/23/82 }
642:S
643:S      { Will be initialized to a call
644:S      to a proc in kernal_initialize.
645:S      To extend error messages, copy old
646:S      proc value into local var and put in
647:S      your new proc ( type errlnk_type ).

```

```

648:S
649:S      Your proc should see if it can handle
650:S      code and return the error msg string,
651:S      if it can't it should call your
652:D      -816 1      local link.
653:S
654:S
655:D      1  PROCEDURE io_escape ( my_code : INTEGER ;
656:D      -816 2      select_code: INTEGER);
657:D      -816 1  FUNCTION io_find_isc ( iod_temp : ANYPTR ) : io_byte;
658:S
659:S

```



```

660:D -816 1 $PAGE$
661:D -816 1 IMPLEMENT
662:S
663:D 1 PROCEDURE io_escape ( my_code : INTEGER ;
664:D 2 select_code: INTEGER);
665:C 2 BEGIN
666:C 2 ioe_isc := select_code;
667:C 2 ioe_result := my_code;
668:C 2 ESCAPE(ioescapecode);
669:C 2 END; { of io_escape }
670:S
671:S
672:D 1 FUNCTION io_find_isc ( iod_temp : ANYPTR ): io_byte;
673:D 2 VAR my_ptr : pio_tmp_ptr;
674:C -4 2 BEGIN
675:C 2 IF iod_temp = NIL
676:C 3 THEN BEGIN
677:C 3 io_find_isc := no_isc ;
678:C 3 END
679:C 3 ELSE BEGIN
680:C 3 my_ptr := iod_temp;
681:C 3 io_find_isc := my_ptr^.my_isc.
682:C 3 END; { of IF }
683:C 2 END; { of io_find_isc }
684:S
685:C 1 END; { of iodeclarations }

```

```

686:D 1 $PAGE$
687:D 1 EXTERNAL MODULE iocomasm ;
688:S
689:S (
690:S date 10/29/81
691:S update 10/29/81
692:S
693:S purpose This module contains the iocomasm code
694:S { binary functions & dma control }
695:D 1 )
696:S
697:D 1 IMPORT iodeclarations;
698:S
699:D 1 EXPORT
700:S
701:D 1 FUNCTION dma_request ( temp : ANYPTR ) : INTEGER;
702:D 1 PROCEDURE dma_release ( temp : ANYPTR );
703:D 1 FUNCTION bit_set ( v : INTEGER ;
704:D 2 b : INTEGER ) : BOOLEAN ;
705:D 1 FUNCTION binand ( x : INTEGER ;
706:D 2 y : INTEGER ) : INTEGER ;
707:D 1 FUNCTION binior ( x : INTEGER ;
708:D 2 y : INTEGER ) : INTEGER ;
709:D 1 FUNCTION bineor ( x : INTEGER ;
710:D 2 y : INTEGER ) : INTEGER ;
711:D 1 FUNCTION bincmp ( x : INTEGER ) : INTEGER ;
712:S
713:S
714:D 1 END; { of iocomasm }

```



```

715:D      1 $PAGE$
716:D      1 {*****}
717:D      1 {*}
718:D      1 {*}
719:D      1 {*      GENERAL GROUP      GENERAL_0}
720:D      1 {*}
721:D      1 {*}
722:D      1 {*****}
723:S
724:S
725:S
726:D      1 MODULE general_0 ;
727:S
728:S      {
729:S          date 07/15/81
730:S          update 03/05/84
731:S
732:S          purpose This module contains the LEVEL 0 GENERAL GROUP procedures.
733:D      1      }
734:S
735:S
736:D      1 IMPORT iodeclarations;
737:S
738:D      1 EXPORT
739:S
740:D      1 VAR
741:D      1 { driver tables }
742:D      -120 1 kbd_crt_drivers : drv_table_type;
743:D      -240 1 dummy_drivers : drv_table_type;
744:S
745:S
746:S
747:D      1 FUNCTION ioread_word ( select_code: type_isc ;
748:D      2      register : io_word )
749:D      -240 2      : io_word ;
750:D      1 PROCEDURE iowrite_word( select_code: type_isc ;
751:D      2      register : io_word ;
752:D      -240 2      value : io_word);
753:D      1 FUNCTION ioread_byte ( select_code: type_isc ;
754:D      2      register : io_word )
755:D      -240 2      : io_byte ;
756:D      1 PROCEDURE iowrite_byte( select_code: type_isc ;
757:D      2      register : io_word ;
758:D      -240 2      value : io_byte);
759:D      1 FUNCTION iostatus ( select_code: type_isc ;
760:D      2      register : io_word )
761:D      -240 2      : io_word ;
762:D      1 PROCEDURE iocontrol ( select_code: type_isc ;
763:D      2      register : io_word ;
764:D      -240 2      value : io_word);
765:D      -240 1 PROCEDURE kernel_initialize;
766:D      -240 1 PROCEDURE io_system_reset;
767:S
768:S
769:D      -240 1 IMPLEMENT
770:S
771:D      -240 1 IMPORT sysglobals ,
772:D      -240 1 iocomasm ;
773:S
774:S

```

```

775:S

```



```

776:D -240 1 $PAGE$
777:S
778:D -240 1 { these are dummy driver procedures used by the kernel }
779:S
780:D 1 PROCEDURE kbd_rdb ( iod_temp : ANYPTR ;
781:D 2 VAR value : CHAR );
782:C 2 BEGIN
783:C 2 IF EOLN(input)
784:C 2 THEN BEGIN
785:C 3 READ(value);
786:C 3 value:=io_carriage_rtn;
787:C 3 END
788:C 3 ELSE BEGIN
789:C 3 READ(value);
790:C 3 END; { of IF EOLN }
791:C 2 END; { of kbd_rdb }
792:S ( tttt TM 9/22/82 )
793:S
794:S
795:D 1 PROCEDURE crt_wtb ( iod_temp : ANYPTR ;
796:D 2 value : CHAR );
797:C 2 BEGIN
798:C 2 WRITE(value);
799:C 2 END; { of crt_wtb }
800:S
801:S
802:S
803:D 1 PROCEDURE simple_init ( temp : ANYPTR );
804:C 2 BEGIN
805:C 2 { this initialization will do nothing }
806:C 2 END; { of simple_init }
807:S
808:D -240 1 { addition of new dummy drivers 0350 TM 8/19/82 }
809:D 1 PROCEDURE dummy_driver( temp : ANYPTR ); { 0350 TM 8/19/82 }
810:C 2 BEGIN { 0350 TM 8/19/82 }
811:C 2 io_escape(ioe_no_driver,io_find_isc(temp));
812:C 2 END;
813:S
814:D 1 PROCEDURE dummy_driver_c( temp : ANYPTR ; dummy : CHAR ); { 0350 TM 8/19/82 }
815:C 2 BEGIN
816:C 2 dummy_driver(temp);
817:C 2 END;
818:S
819:D 1 PROCEDURE dummy_driver_a( temp : ANYPTR ; ANYVAR dummy : ANYPTR ); { 0350 TM 8/19/82 }
820:C 2 BEGIN
821:C 2 dummy_driver(temp);
822:C 2 END;
823:S
824:D 1 PROCEDURE dummy_driver_w( temp : ANYPTR ; dummy : io_word ); { 0350 TM 8/19/82 }
825:C 2 BEGIN
826:C 2 dummy_driver(temp);
827:C 2 END;
828:S
829:D 1 PROCEDURE dummy_driver_dw(temp : ANYPTR ; dummy,d2 : io_word ); { 0350 TM 8/19/82 }
830:C 2 BEGIN
831:C 2 dummy_driver(temp);
832:C 2 END;
833:S
834:D 1 PROCEDURE dummy_driver_wa(temp : ANYPTR ; dummy : io_word ; VAR d2 : io_word ); { 0350 TM 8/19/82 }
835:C 2 BEGIN

```

```

836:C 2 dummy_driver(temp);
837:C 2 END;
838:S
839:D 1 PROCEDURE dummy_driver_b( temp : ANYPTR ; dummy : io_bit ); { 0350 TM 8/19/82 }
840:C 2 BEGIN
841:C 2 dummy_driver(temp);
842:C 2 END;
843:S
844:D 1 PROCEDURE dummy_driver_ba(temp : ANYPTR ; dummy : io_bit ; VAR d2 : 800LEAN ); { 0350 TM 8/19/82 }
845:C 2 BEGIN
846:C 2 dummy_driver(temp);
847:C 2 END;
848:S
849:S

```



```

850:D -240 1 $PAGES
851:S
852:D 1 FUNCTION ioread_word ( select_code: type_isc ;
853:D 2 register : io_word ;
854:D 2 : io_word ;
855:D -4 2 VAR p : ^io_word;
856:C 2 BEGIN
857:C 2 p := ANYPTR(isc_table[select_code].card_ptr);
858:C 2 IF p = NIL
859:C 2 THEN BEGIN
860:C 2 { error }
861:C 2 io_escape(ioe_no_card,select_code);
862:C 2 END
863:C 2 ELSE BEGIN
864:C 2 p:=ANYPTR(INTEGER(p)+register);
865:C 2 ioread_word:=p^;
866:C 2 END; { of IF }
867:C 2 END; { of ioread_word }
868:S
869:S
870:S
871:D 1 PROCEDURE iowrite_word( select_code: type_isc ;
872:D 2 register : io_word ;
873:D 2 value : io_word);
874:D -4 2 VAR p : ^io_word;
875:C 2 BEGIN
876:C 2 p := ANYPTR(isc_table[select_code].card_ptr);
877:C 2 IF p = NIL
878:C 2 THEN BEGIN
879:C 2 { error }
880:C 2 io_escape(ioe_no_card,select_code);
881:C 2 END
882:C 2 ELSE BEGIN
883:C 2 p:=ANYPTR(INTEGER(p)+register);
884:C 2 p^:=value;
885:C 2 END; { of IF }
886:C 2 END; { of iowrite_word }
887:S
888:S
889:S
890:D 1 FUNCTION ioread_byte ( select_code: type_isc ;
891:D 2 register : io_word ;
892:D 2 : io_byte ;
893:D -4 2 TYPE mycharptr = ^CHAR;
894:D 2 VAR p : ^CHAR;
895:C 2 BEGIN
896:C 2 p := ANYPTR(isc_table[select_code].card_ptr);
897:C 2 IF p = NIL
898:C 2 THEN BEGIN
899:C 2 { error }
900:C 2 io_escape(ioe_no_card,select_code);
901:C 2 END
902:C 2 ELSE BEGIN
903:C 2 ioread_byte:=ORD(mycharptr(ADDR(p^,register))^);
904:C 2 END; { of IF }
905:C 2 END; { of ioread_word }
906:S
907:S
908:S
909:D 1 PROCEDURE iowrite_byte( select_code: type_isc ;

```

```

910:D 2 register : io_word ;
911:D 2 value : io_byte);
912:D -4 2 TYPE mycharptr = ^CHAR;
913:D 2 VAR p : ^CHAR;
914:C 2 BEGIN
915:C 2 p := ANYPTR(isc_table[select_code].card_ptr);
916:C 2 IF p = NIL
917:C 2 THEN BEGIN
918:C 2 { error }
919:C 2 io_escape(ioe_no_card,select_code);
920:C 2 END
921:C 2 ELSE BEGIN
922:C 2 p:=ANYPTR(INTEGER(p)+register);
923:C 2 p^:=CHR(value);
924:C 2 END; { of IF }
925:C 2 END; { of iowrite_byte }
926:S
927:S
928:S
929:D 1 FUNCTION iostatus ( select_code: type_isc ;
930:D 2 register : io_word ;
931:D 2 : io_word ;
932:D -2 2 VAR value : io_word;
933:C 2 BEGIN
934:C 2 WITH isc_table[select_code] DD
935:C 2 CALL(io_drv_ptr^.iod_fds,
936:C 2 io_imp_ptr,
937:C 2 register,
938:C 2 value);
939:C 2 iostatus:=value;
940:C 2 END; { of iostatus }
941:S
942:S
943:S
944:D 1 PROCEDURE iocontrol ( select_code: type_isc ;
945:D 2 register : io_word ;
946:D 2 value : io_word);
947:D -2 2 VAR my_value : io_word;
948:C 2 BEGIN
949:C 2 my_value:=value;
950:C 2 WITH isc_table[select_code] DD
951:C 2 CALL(io_drv_ptr^.iod_wtc,
952:C 2 io_imp_ptr,
953:C 2 register,
954:C 2 my_value);
955:C 2 END; { of iocontrol }
956:S

```



```
957:D -240 1 $SPACE$
958:D 1 PROCEDURE end_error_link ( errorcode : INTEGER ;           { 0082 TM 7/23/82 }
959:D 2                               VAR s : io_string );           { 0082 TM 7/23/82 }
960:C 2 BEGIN                                                       { 0082 TM 7/23/82 }
961:C 2   s := 'unrecognized error';                                   { 0082 TM 7/23/82 }
962:C 2 END; { of end_error_link }                                   { 0082 TM 7/23/82 }
963:S
964:D 1 PROCEDURE kernel_initialize;
965:D 2   VAR io_isc : type_isc;
966:D 2       dc_dummy : INTEGER;
967:D 2       dummy : INTEGER;
968:D 2       double : BOOLEAN; { indicates an int. that takes 2 s.c. }
969:D 2       bigtemp : ^io_temp_type2;
970:D 2       bigtemp3 : ^io_temp_type3;
971:D 2
972:C 2 BEGIN
973:C 2
974:S
975:C 2   io_revid := '10 3.0: '; { io library revision/id - jws 8/03/83 }
976:S
977:C 2   io_error_link := end_error_link; { error msg extension } { 0082 TM 7/23/82 }
978:S
979:C 2   double := FALSE;
980:S
981:S
982:C 2   { determine what interfaces are present }
983:S
984:C 2   FOR io_isc:=iominisc TO iomaxisc DO WITH isc_table[io_isc] DO BEGIN
985:C 4     user_time :=0;
986:C 4     card_id := no_id;
987:C 4     IF io_isc<minrealisc
988:C 5     THEN BEGIN
989:C 5       card_type := system_card;
990:C 5       io_tmp_ptr := NIL;
991:C 5       CASE io_isc OF
992:C 6         1: BEGIN
993:C 6           card_id:=internal_crt; { BUG aaaa TM 5/24/82 }
994:C 6           END;
995:C 6         2: BEGIN
996:C 6           card_id:=internal_kbd; { BUG aaaa TM 5/24/82 }
997:C 6           END;
998:C 6         3: BEGIN
999:C 6           card_ptr := ANYPTR(5242880); { dma address $500000 }
1000:C 6           { let DMA driver module hunt for dma }
1001:C 6           dma_here := FALSE;
1002:C 6           END; { of BEGIN }
1003:C 6
1004:C 6           OTHERWISE BEGIN ENO; { other internal interfaces }
1005:C 6         END; { of CASE io_isc }
1006:S
1007:S
1008:C 5   END
1009:C 5   ELSE BEGIN
1010:C 5     card_type:=no_card;
1011:C 5     IF (io_isc=7) and not(sysflag.nointphib) { jws2 6/28/83 }
1012:C 5     THEN BEGIN
1013:C 5
1014:C 5
1015:C 5
1016:C 5
```

```
1017:C 6   NEW(io_tmp_ptr); { get temp space }
1018:C 6   card_type:=hpib_card;
1019:C 6   card_id :=internal_hpib;
1020:C 6   card_ptr :=ANYPTR(HEX('478000'));
1021:C 6   ENO
1022:C 6   ELSE BEGIN
1023:C 6     IF double
1024:C 7     THEN BEGIN
1025:C 7       double := FALSE;
1026:C 7
1027:S
1028:C 7     END
1029:C 7     ELSE BEGIN
1030:C 7       card_ptr := ANYPTR(HEX('600000')+((io_isc MOD 32)*65536));
1031:C 7       TRY
1032:C 8         dummy := ioread_byte(io_isc,1);
1033:C 8         dummy := dummy MOD 128; { mask off remote id bit }
1034:S
1035:C 8         IF dummy <= 8 THEN CASE dummy OF
1036:S
1037:C 10           1: BEGIN
1038:C 10             card_type:=hpib_card;
1039:C 10             card_id :=hp98824;
1040:C 10             NEW(io_tmp_ptr); { get temp space }
1041:C 10             END;
1042:C 10
1043:C 10           2: BEGIN
1044:C 10             card_type:=serial_card;
1045:C 10             card_id :=hp98825;
1046:C 10             NEW(bigtemp3); {jw 7/12/82}
1047:C 10             io_tmp_ptr := ANYPTR(bigtemp3); { get temp space }
1048:C 10             {jw 7/12/82}
1049:C 10             END;
1050:S
1051:C 10           3: BEGIN
1052:C 10             card_type:=gpio_card;
1053:C 10             card_id :=hp98822;
1054:C 10             NEW(io_tmp_ptr); { get temp space }
1055:C 10             END;
1056:S
1057:S
1058:C 10           8: BEGIN
1059:C 10             card_type:=hpib_card;
1060:C 10             card_id :=hp98825;
1061:C 10             NEW(io_tmp_ptr); { get temp space }
1062:C 10             END;
1063:S
1064:C 10           OTHERWISE BEGIN
1065:C 10             NEW(io_tmp_ptr); { get temp space }
1066:C 10             card_type:=other_card;
1067:C 10             { note - this card will get TEMP space }
1068:C 10             END;
1069:S
1070:C 10           ENO { of CASE }
1071:C 10           ELSE BEGIN
1072:S
1073:C 9             IF dummy <= 30 THEN { jws 3/25/83 }
1074:C 9             CASE dummy of { jws 3/25/83 }
1075:C 10
1076:S
```



```

1077:C      11      25: BEGIN                                { jws3 2/09/84 }
1078:C      11      card_id:=hpGATOR;                      { jws3 2/09/84 }
1079:C      11      card_type:=graphics_card;              { jws3 2/09/84 }
1080:C      11      NEW(io_tmp_ptr);                        { jws3 2/09/84 }
1081:C      11      END;                                    { jws3 2/09/84 }
1082:S
1083:C      11
1084:C      11      27: BEGIN                                { jws 3/25/83 }
1085:C      11      card_id:=hp98253;                        { jws 3/25/83 }
1086:C      11      card_type:=eprom_prgrmr;                { jws 3/25/83 }
1087:C      11      NEW(io_tmp_ptr);                        { jws 3/25/83 }
1088:S      11      END;                                    { jws 3/25/83 }
1089:C      11
1090:C      11      28: BEGIN
1091:C      11      card_id := hp98627;
1092:C      11      card_type := graphics_card;
1093:C      11      double := TRUE;
1094:C      11      NEW(io_tmp_ptr);                        ( get temp space )
1095:S      11      END;
1096:C      11
1097:C      11      29: BEGIN
1098:C      11      ( id=28 is also set aside for double wide cards )
1099:C      11      double := TRUE;
1100:C      11      NEW(bigtemp3);                                { jws 3/25/83 }
1101:C      11      io_tmp_ptr:=ANYPTR(bigtemp3);              { jws 3/25/83 }
1102:C      11      card_type:=other_card;                    { jws 3/25/83 }
1103:S      11      END;
1104:C      11
1105:C      11      30: BEGIN                                { jws 3/25/83 }
1106:C      11      card_id:=hp98259;                        { jws 3/25/83 }
1107:C      11      card_type:=bubble_card;                    { jws 3/25/83 }
1108:C      11      NEW(bigtemp);                                { jws 3/25/83 }
1109:C      11      io_tmp_ptr:=ANYPTR(bigtemp);              { jws 3/25/83 }
1110:S      11      END;
1111:C      11      OTHERWISE                                { jws 3/25/83 }
1112:C      11      BEGIN                                        { jws 3/25/83 }
1113:C      11      NEW(bigtemp3);                                { jws 3/25/83 }
1114:C      11      io_tmp_ptr:=ANYPTR(bigtemp3);              { jws 3/25/83 }
1115:C      11      card_type:=other_card;                    { jws 3/25/83 }
1116:C      11      END;                                        { jws 3/25/83 }
1117:S
1118:C      11      END; ( of case )                            ( jws 3/25/83 )
1119:S
1120:S
1121:S
1122:C      9      IF dummy=52
1123:C      10      THEN BEGIN
1124:C      10      dc_dummy := ioread_byte(io_isc,16395)*256+ioread_byte(io_isc,16393);
1125:C      10      IF dc_dummy < 32768
1126:C      11      THEN BEGIN
1127:C      11      dc_dummy := ioread_byte(io_isc,dc_dummy*2+1);
1128:C      11      IF (dc_dummy MOD 128) = 1
1129:C      12      THEN BEGIN
1130:C      12      card_id := hp_datacomm;                        ( since I don't know yet )
1131:C      12      card_type := Serial_card;
1132:C      12      END; ( of IF )
1133:C      11      END; ( of IF )
1134:C      10      NEW(bigtemp);
1135:C      10      io_tmp_ptr:=ANYPTR(bigtemp);
1136:C      10      END; ( of BEGIN )

```

```

1137:S
1138:C      9      IF dummy=66
1139:C      10      THEN BEGIN                                { jws4 3/5/84 }
1140:C      10      card_id:=hp98644;                        { jws4 3/5/84 }
1141:C      10      card_type:=serial_card;                  { jws4 3/5/84 }
1142:C      10      NEW(bigtemp3);                          { jws4 3/5/84 }
1143:C      10      io_tmp_ptr:=ANYPTR(bigtemp3);            { jws4 3/5/84 }
1144:C      10      END;                                    { jws4 3/5/84 }
1145:S
1146:C      9
1147:C      10      IF ( dummy > 30 ) AND                      ( jws4 3/5/84 )
1148:C      10      ( dummy <= 52 ) AND ( dummy<>66 )
1149:C      10      THEN BEGIN
1150:C      10      NEW(bigtemp3);                                { TM 8/20/82 }
1151:C      10      io_tmp_ptr := ANYPTR(bigtemp3);          { TM 8/20/82 }
1152:C      10      ( note - this card WILL get LARGE temp space )
1153:C      10      card_type:=other_card;
1154:S      10      END;
1155:C      9      END; ( of IF dummy <= 8 )
1156:S
1157:C      8      RECOVER
1158:C      8      BEGIN
1159:C      8      IF ( escapecode=-11 ) OR ( escapecode=-12 )
1160:C      9      THEN BEGIN
1161:C      9      ( no card at this address )
1162:C      8      END
1163:C      8      ELSE BEGIN
1164:C      8      ( some other problem )
1165:C      9      ESCAPE(escapecode);
1166:C      9      END;
1167:C      8      END; ( of RECOVER BEGIN )
1168:S
1169:S
1170:C      7      END; ( of IF double )
1171:C      6      END; ( of IF io_isc=7 )
1172:C      5      END; ( of IF io_isc<minrealisc THEN/ELSE )
1173:C      4      END; ( of FOR WITH isc_table[io_isc] DO BEGIN )
1174:S
1175:S
1176:C      2      ( set up the driver tables )                { 0350 TM 8/18/82 }
1177:C      2      WITH dummy_drivers DO BEGIN                { 0350 TM 8/19/82 }
1178:C      3      iod_init := ISRPROTYPE (dummy_driver);    { 0350 TM 8/18/82 }
1179:C      3      iod_rdb := io_proc_vc (dummy_driver_a);
1180:C      3      iod_wtb := dummy_driver_c;
1181:C      3      iod_rdw := io_proc_vw (dummy_driver_a);
1182:C      3      iod_wtw := dummy_driver_w;
1183:C      3      iod_rds := io_proc_vs (dummy_driver_wa);
1184:C      3      iod_wtc := dummy_driver_dw;
1185:C      3      iod_end := io_proc_vb (dummy_driver_a);
1186:C      3      iod_tfr := io_proc_ptr (dummy_driver_a);
1187:C      3      iod_send := dummy_driver_c;
1188:C      3      iod_ppoll := io_proc_vc (dummy_driver_a);
1189:C      3      iod_set := dummy_driver_b;
1190:C      3      iod_clr := dummy_driver_b;
1191:C      3      iod_test := dummy_driver_ba;
1192:C      3      END; ( of WITH )
1193:S
1194:C      3      { 0350 TM 8/19/82 }
1195:S
1196:S
1197:S
1198:S
1199:S

```



```

1197:S
1198:C      2      { initialize the temp space }
1199:C      2      FOR io_isc := iominisc TO iomaxisc DO BEGIN
1200:S
1201:C      3      { set up dummy drivers for the interfaces }
1202:C      3      isc_table[io_isc].io_drv_ptr:=ADDR(dummy_drivers);
1203:S
1204:C      3      IF isc_table[io_isc].io_tmp_ptr <> NIL
1205:C      4      THEN WITH isc_table[io_isc].io_tmp_ptr^ DO BEGIN
1206:C      5          card_addr := isc_table[io_isc].card_ptr;
1207:C      5          eirbyte  := CHR( 0 );
1208:C      5          my_isc    := io_isc ;
1209:C      5          timeout   := 0 ;
1210:C      5          addressed := -1;
1211:C      5          in_bufptr := NIL ;
1212:C      5          out_bufptr:= NIL ;
1213:C      5          user_isr.dummy_sl := NIL;
1214:C      5          user_isr.dummy_pr := NIL;
1215:C      5          user_parm    := NIL;
1216:C      5          myisrib.INTREGADDR:= NIL;
1217:C      5      END; { of IF WITH io_tmp_ptr^ DO }
1218:C      3      END; { of FOR DO BEGIN }
1219:S
1220:C      2      dma_isrib0.INTREGADDR := NIL;
1221:C      2      dma_isrib1.INTREGADDR := NIL;
1222:S
1223:S
1224:S      { note - because of the ISRs - this routine can only be called
1225:S      once - at INITLIB time. If it is called again after that
1226:S      invocation the ISR structure will be in very bad shape and
1227:S      will probably hang the machine }
1228:S      2
1229:S
1230:S
1231:C      2      kbd_crt_drivers:=dummy_drivers;
1232:C      2      WITH kbd_crt_drivers DO BEGIN
1233:C      3          iod_rdb := kbd_rdb;
1234:C      3          iod_wtb := crt_wtb;
1235:C      3          iod_init := simple_init;
1236:C      3      END; { of WITH }
1237:S
1238:S
1239:C      2      isc_table[1].io_drv_ptr := ADDR(kbd_crt_drivers);
1240:C      2      isc_table[2].io_drv_ptr := ADDR(kbd_crt_drivers);
1241:S
1242:S
1243:C      2      { set up clear i/o hook to do an io system reset }
1244:C      2      cleariohook := io_system_reset;
1245:S
1246:S
1247:S
1248:C      2      END; { of kernel_initialize }
1249:S

```

```

1250:D      -240  1  $PAGE$
1251:S
1252:D      1  PROCEDURE io_system_reset;
1253:D      -2  2  VAR io_isc : Type_isc;
1254:C      2  BEGIN
1255:S
1256:C      2      { initialize the interfaces }
1257:S
1258:C      2      FOR io_isc:=iominisc TO iomaxisc DO
1259:C      3      WITH isc_table[io_isc] DO BEGIN
1260:S
1261:C      4          user_time := 0;
1262:S
1263:C      4          IF io_tmp_ptr <> NIL
1264:C      5          THEN WITH io_tmp_ptr^ DO BEGIN
1265:C      6              eirbyte := CHR( 0 );
1266:C      6              my_isc  := io_isc ;
1267:C      6              timeout  := 0 ;
1268:C      6              user_isr.dummy_sl := NIL;
1269:C      6              user_isr.dummy_pr := NIL;
1270:C      6              user_parm    := NIL;
1271:C      6              END; { of IF THEN WITH BEGIN }
1272:S
1273:C      4      END; { of FOR WITH BEGIN }
1274:S
1275:S      { these two FOR blocks are separate in case two HPB interfaces are
1276:C      2      connected in one machine - no funny user isr's will happen. }
1277:S
1278:C      2      FOR io_isc:=iominisc TO iomaxisc DO
1279:C      3      WITH isc_table[io_isc] DO BEGIN
1280:S
1281:C      4          IF io_drv_ptr^.iod_init <> dummy_driver
1282:C      5          THEN BEGIN
1283:C      5              CALL ( io_drv_ptr^.iod_init ,
1284:C      5                  io_tmp_ptr );
1285:C      5              END; { of IF }
1286:S
1287:C      4      END; { of FOR WITH BEGIN }
1288:S
1289:S
1290:S      { In case - for some messed up reason ( typically this is when the user
1291:S      doesn't call ioinitalize/iouninitialize ) - the dma resources are not
1292:C      2      relinquished by the init routines then this will free the resources. }
1293:S
1294:C      2      IF dma_isc 0 <> no_isc
1295:C      3      THEN BEGIN
1296:C      3          dma_release(isc_table[dma_isc_0] io_tmp_ptr);
1297:C      3      END; { of IF }
1298:S
1299:C      2      IF dma_isc 1 <> no_isc
1300:C      3      THEN BEGIN
1301:C      3          dma_release(isc_table[dma_isc_1] io_tmp_ptr);
1302:C      3      END; { of IF }
1303:S
1304:S
1305:S
1306:C      2      END; { of io_system_reset }
1307:S
1308:S
1309:S

```



```
1310:S
1311:C      1 END;      { of general_0 }
```

```
1312:D      1 $PAGE$
1313:D      1 {*****}
1314:D      1 { * }
1315:D      1 { * }
1316:D      1 { * IOLIBRARY_KERNEL }
1317:D      1 { * }
1318:D      1 { * }
1319:D      1 {*****}
1320:S
1321:S
1322:S
1323:D      1 IMPORT general_0 ,
1324:D      1          LOADER ;
1325:S
1326:S
1327:C      1 BEGIN
1328:S
1329:C      1 kernel initialize;
1330:C      1 MARKUSER;
1331:C      1 END; { of iolibrary_kernel }
1332:S
```

No errors. No warnings.

***** Nonstandard language features enabled *****

KEYS

Description

KEYS contains the initialization routine for the keyboard, the keyboard TM, the general translation routine (executed via KBDTRANSHOOK in A804XDVR), keyboard and RPG ISRs.

Requirements

SYSGLOBALS, ASM, MISC, and SYSDEVS.

Notes

Some of the quoted strings in the following listing (Roman Extension characters, etc.) are printed incorrectly due to the fact that the CRT character set does not match the printer's character set.

See also A804XDVR for calls to some of these routines.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1: (*
2:
3:
4:      (-) Copyright Hewlett-Packard Company, 1984.
5:      All rights are reserved. Copying or other
6:      reproduction of this program except for archival
7:      purposes is prohibited without the prior
8:      written consent of Hewlett-Packard Company.
9:
10:
11:      RESTRICTED RIGHTS LEGEND
12:
13:      Use, duplication, or disclosure by the Government
14:      is subject to restrictions as set forth in
15:      paragraph (b) (3) (8) of the Rights in Technical
16:      Data and Computer Software clause in
17:      DAR 7-104.9(a).
18:
19:      HEWLETT-PACKARD COMPANY
20:      Fort Collins, Colorado *)
21:
22: 0 $MODCAL $
23: 0 $heap_dispose off$
24: 0 $iocheck off$
25: 0 $range off$ $ovflcheck off$
26: 0 $debug off$
27: 0 $STACKCHECK OFF$
28:
29: 0 $SEARCH 'INITLOAD', 'ASM', 'INIT', 'SYSDEVS'$
30:
31: 0 program keysinit;
32:
33: 1 module keys;
34: 1 import sysglobals, asm, misc, sysdevs;
35: 1 export
36: 1 procedure initkeys;
37:
38: 1 implement
39: 1 const
40: 1 sysnorm =
41: 1 string80[' k0 |RECALL |CLR-END| CONT | | STEP | ALPHA | GRAPH | k9 |'];
42: 1 sysshift =
43: 1 string80[' k0 |RECALL |CLR-END| CONT | | ANYCHAR | DMP A | DMP G | k9 |'];
44: 1 type
45: 1 menu1 = array[boolean] of menutype;
46: 1 menu2 = array[m_none..m_sysshift] of menu1;
47: 1 const
48: 1 mstates = menu2[
49: 1 { no menu } menu1[m_sysnorm,m_sysshift],
50: 1 { normal } menu1[m_none,m_sysshift],
51: 1 { shifted } menu1[m_sysnorm,m_none]];
52: 1 var
53: -1 1 anychar : boolean;
54: -4 1 buildchar : shortint;
55: -6 1 buildcount : 0..4;
56: -14 1 anycharsavehook : kbdhooktype;
57:
58:
59: 1 procedure stopaction;
60: 2 begin

```

```

61: 2 actionspending := 0; escape(-20);
62: 2 end;
63:
64: 1 procedure cntrlpausekey;
65: 2 type strin = string[1];
66: 2 const qm = strin['?'];
67: ***WARNING: (line 67): ADDR of a constant may not be supported on other implementations
68: 2 begin call(debugger,4,integer(addr(qm)),0) end;
69:
70: -14 1 procedure pausekey;
71: 2 begin call(debugger,6,0,0) end;
72:
73: -14 1 procedure dokbdio(fp: fbp; request: amrequesttype; anyvar buffer: window;
74: 2 length, position: integer);
75: -4 2 var interruptlevel: integer;
76: -5 2 var commandinprogress: char;
77: -6 2 zchr: char;
78: -10 2 buf: charptr;
79: 2 begin
80: 2 iresult := ord(inoerror);
81: 2 buf := addr(buffer);
82: 2 case request of
83: 3 flush: (do nothing);
84: 3 unitstatus: fp^.fbusy := keybuffer^.size = 0;
85: 3 clearunit: begin
86: 4 interruptlevel := intlevel;
87: 4 if interruptlevel < 1 then setintlevel(1);
88: 4 keybufops(kclear,zchr); ( clear typeahead buffer )
89: 4 setintlevel(interruptlevel);
90: 4 end;
91: 3
92: 3 if false$
93: 3 writeeol,
94: 3 startwrite,
95: 3 writebytes: crtio(fp, request, buffer, length, 0);
96: 3 $ends$
97: 3
98: 3 readtoeol,
99: 3 readbytes,
100: 3 startread:
101: 3 begin
102: 4 if request = readtoeol then
103: 4 begin
104: 5 buf := addr(buf^, 1); ( format buffer as a string )
105: 5 buffer[0] := chr(0);
106: 4 end;
107: 4 while length > 0 do
108: 4 begin
109: 5 if runlight <> chr(idle) then commandinprogress := runlight;
110: 5 interruptlevel := intlevel;
111: 5 with keybuffer^ do
112: 6 repeat
113: 7 while size = 0 do call(kbdwaithook); ( wait for keys )
114: 7 setintlevel(7); (disable interrupts)
115: 7 if size = 0 then setintlevel(interruptlevel);
116: 6 until size > 0;
117: 6 setrunlight(commandinprogress);
118: 4 with keybuffer^ do buf^:=buffer^[outp]; ( BUG FIX #64 )
119: 4

```



```

120:C      4      if buf^ = chr(etc) then length := 0
121:C      5              else length := length-1;
122:C      4      if (buf^=eol) and (request=readtoeol) then length := 0
123:C      5      else
124:C      6      begin
125:C      7      keybufops(kgetchar,buf^); { get next character from input }
126:C      8      fp^.feoln := false; { BUG FIX #84 }
127:C      5      fp^.feoln := false; { set not end of line }
128:C      5      buf := addr(buf^, 1); { increment output buffer address }
129:C      5      if request = readtoeol then buffer[0] := chr(ord(buffer[0])+1); { increment string size }
130:C      5      end;
131:C      5      setintlevel(interruptlevel); { restore interrupt level }
132:C      4      end; { while }
133:C      4      end; { while }
134:C      3      if request = startread then call(fp^.feot, fp);
135:C      3      end;
136:C      3      otherwise ioresult := ord(ibadrequest);
137:C      3      end;
138:C      2      end; { dokbdio }
139:C      2      end;
140:C      1      end;
141:C      1      procedure nonadvkeys;
142:C      2      type
143:C      3      aa = packed array[' ' .. ' ' of char;
144:C      3      const
145:C      4      ala = aa['DH@Lb']; { tables modified/widened 5/7/84 SFB/RQ }
146:C      4      ale = aa['E@NMe'];
147:C      4      ali = aa['UYQ@'];
148:C      4      alo = aa['FISN'];
149:C      4      alu = aa['GKCOU'];
150:C      4      aua = aa['^Xa'];
151:C      4      aue = aa['^#SE'];
152:C      4      auu = aa['ef&'];
153:C      4      auo = aa['gh_Z'];
154:C      4      auu = aa['m-[U'];
155:C      4      var
156:C      5      nc : char;
157:C      5      -1      begin
158:C      6      nc := keybuffer^.non_char;
159:C      6      if langtable[langindex]^can_nonadv and (nc<>' ') then
160:C      7      begin
161:C      8      if (langcom.key=rightchar) or (langcom.key=' ') then langcom.key:=nc
162:C      9      else
163:C      10     case langcom.key of
164:C      11     'a': langcom.key := ala[nc];
165:C      12     'e': langcom.key := ale[nc];
166:C      13     'i': langcom.key := ali[nc];
167:C      14     'o': langcom.key := alo[nc];
168:C      15     'u': langcom.key := alu[nc];
169:C      16     'n': if nc = '^' then langcom.key := '7';
170:C      17     'y': if nc = '^' then langcom.key := 'o'; { SFB/RQ 5/7/84 }
171:C      18     'A': langcom.key := aua[nc];
172:C      19     'E': langcom.key := aue[nc];
173:C      20     'I': langcom.key := auu[nc];
174:C      21     'O': langcom.key := auo[nc]; { SFB/RQ 5/7/84 }
175:C      22     'U': langcom.key := auu[nc];
176:C      23     'N': if nc = '^' then langcom.key := '6';
177:C      24     'Y': if nc = '^' then langcom.key := 'n'; { SFB/RQ 5/7/84 }
178:C      25     otherwise
179:C      26     end;

```

```

180:C      5      end; { case key }
181:C      3      keybuffer^.non_char := ' ';
182:C      3      end;
183:C      2      end; { nonadvkeys }
184:C      1      end;
185:C      1      procedure generaltrans;
186:C      2      { multi lingual translate procedure }
187:C      3      { all languages except katakana }
188:C      3      label 1;
189:C      3      -1      var tshift : boolean;
190:C      3      begin
191:C      4      if langcom.status=0 then
192:C      5      begin { check key for non advanced key involvement }
193:C      6      if (langcom.key>=#168) and (langcom.key<=#172) then
194:C      7      langcom.result:=nonadv_key
195:C      8      else
196:C      9      begin langcom.result:=alpha_key; nonadvkeys; end;
197:C      10     end
198:C      11     else { process key code }
199:C      12     if langcom.data>127 then langcom.result:=ignored_key
200:C      13     else
201:C      14     with langcom, langtable[langindex]^, keytable[data] do
202:C      15     begin
203:C      16     result := keyclass;
204:C      17     if result=ignored_key then goto 1;
205:C      18     extension := extension and not no_extension;
206:C      19     if extension then
207:C      20     begin { call alternate semantics routine }
208:C      21     extension:=false; langindex:=1-langindex; { 0->1 or 1->0 }
209:C      22     call(langtable[langindex]^semantics);
210:C      23     langindex:=1-langindex;
211:C      24     end
212:C      25     else
213:C      26     begin { normal processing }
214:C      27     control:= control and not no_control;
215:C      28     shift := shift and not no_shift;
216:C      29     tshift := (kdcapslock and not no_capslock) <> shift;
217:C      30     key := keys[tshift];
218:C      31     if (result=alpha_key) and control then key := chr(ord(key) mod 32);
219:C      32     if result=alpha_key then nonadvkeys ;
220:C      33     { S/21/84 SFB }
221:C      34     else
222:C      35     if result=special_key then
223:C      36     if (not tshift and not control and (key=rightchar)) then
224:C      37     begin result:=alpha_key; nonadvkeys; end; { 5/9/84 RQ/SFB }
225:C      38     end;
226:C      39     end; { with langcom etc }
227:C      40     1: end; { generaltrans }
228:C      41     end;
229:C      42     end;
230:C      43     end;
231:C      44     end;
232:C      45     end;
233:C      46     end;
234:C      47     end;
235:C      48     end;
236:C      49     end;
237:C      50     end;
238:C      51     end;
239:C      52     end;
240:C      53     end;
241:C      54     end;
242:C      55     end;
243:C      56     end;
244:C      57     end;
245:C      58     end;
246:C      59     end;
247:C      60     end;
248:C      61     end;
249:C      62     end;
250:C      63     end;
251:C      64     end;
252:C      65     end;
253:C      66     end;
254:C      67     end;
255:C      68     end;
256:C      69     end;
257:C      70     end;
258:C      71     end;
259:C      72     end;
260:C      73     end;
261:C      74     end;
262:C      75     end;
263:C      76     end;
264:C      77     end;
265:C      78     end;
266:C      79     end;
267:C      80     end;
268:C      81     end;
269:C      82     end;
270:C      83     end;
271:C      84     end;
272:C      85     end;
273:C      86     end;
274:C      87     end;
275:C      88     end;
276:C      89     end;
277:C      90     end;
278:C      91     end;
279:C      92     end;
280:C      93     end;
281:C      94     end;
282:C      95     end;
283:C      96     end;
284:C      97     end;
285:C      98     end;
286:C      99     end;
287:C      100    end;
288:C      101    end;
289:C      102    end;
290:C      103    end;
291:C      104    end;
292:C      105    end;
293:C      106    end;
294:C      107    end;
295:C      108    end;
296:C      109    end;
297:C      110    end;
298:C      111    end;
299:C      112    end;
300:C      113    end;
301:C      114    end;
302:C      115    end;
303:C      116    end;
304:C      117    end;
305:C      118    end;
306:C      119    end;
307:C      120    end;
308:C      121    end;
309:C      122    end;
310:C      123    end;
311:C      124    end;
312:C      125    end;
313:C      126    end;
314:C      127    end;
315:C      128    end;
316:C      129    end;
317:C      130    end;
318:C      131    end;
319:C      132    end;
320:C      133    end;
321:C      134    end;
322:C      135    end;
323:C      136    end;
324:C      137    end;
325:C      138    end;
326:C      139    end;
327:C      140    end;
328:C      141    end;
329:C      142    end;
330:C      143    end;
331:C      144    end;
332:C      145    end;
333:C      146    end;
334:C      147    end;
335:C      148    end;
336:C      149    end;
337:C      150    end;
338:C      151    end;
339:C      152    end;
340:C      153    end;
341:C      154    end;
342:C      155    end;
343:C      156    end;
344:C      157    end;
345:C      158    end;
346:C      159    end;
347:C      160    end;
348:C      161    end;
349:C      162    end;
350:C      163    end;
351:C      164    end;
352:C      165    end;
353:C      166    end;
354:C      167    end;
355:C      168    end;
356:C      169    end;
357:C      170    end;
358:C      171    end;
359:C      172    end;
360:C      173    end;
361:C      174    end;
362:C      175    end;
363:C      176    end;
364:C      177    end;
365:C      178    end;
366:C      179    end;
367:C      180    end;
368:C      181    end;
369:C      182    end;
370:C      183    end;
371:C      184    end;
372:C      185    end;
373:C      186    end;
374:C      187    end;
375:C      188    end;
376:C      189    end;
377:C      190    end;
378:C      191    end;
379:C      192    end;
380:C      193    end;
381:C      194    end;
382:C      195    end;
383:C      196    end;
384:C      197    end;
385:C      198    end;
386:C      199    end;
387:C      200    end;
388:C      201    end;
389:C      202    end;
390:C      203    end;
391:C      204    end;
392:C      205    end;
393:C      206    end;
394:C      207    end;
395:C      208    end;
396:C      209    end;
397:C      210    end;
398:C      211    end;
399:C      212    end;
400:C      213    end;
401:C      214    end;
402:C      215    end;
403:C      216    end;
404:C      217    end;
405:C      218    end;
406:C      219    end;
407:C      220    end;
408:C      221    end;
409:C      222    end;
410:C      223    end;
411:C      224    end;
412:C      225    end;
413:C      226    end;
414:C      227    end;
415:C      228    end;
416:C      229    end;
417:C      230    end;
418:C      231    end;
419:C      232    end;
420:C      233    end;
421:C      234    end;
422:C      235    end;
423:C      236    end;
424:C      237    end;
425:C      238    end;
426:C      239    end;
427:C      240    end;
428:C      241    end;
429:C      242    end;
430:C      243    end;
431:C      244    end;
432:C      245    end;
433:C      246    end;
434:C      247    end;
435:C      248    end;
436:C      249    end;
437:C      250    end;
438:C      251    end;
439:C      252    end;
440:C      253    end;
441:C      254    end;
442:C      255    end;
443:C      256    end;
444:C      257    end;
445:C      258    end;
446:C      259    end;
447:C      260    end;
448:C      261    end;
449:C      262    end;
450:C      263    end;
451:C      264    end;
452:C      265    end;
453:C      266    end;
454:C      267    end;
455:C      268    end;
456:C      269    end;
457:C      270    end;
458:C      271    end;
459:C      272    end;
460:C      273    end;
461:C      274    end;
462:C      275    end;
463:C      276    end;
464:C      277    end;
465:C      278    end;
466:C      279    end;
467:C      280    end;
468:C      281    end;
469:C      282    end;
470:C      283    end;
471:C      284    end;
472:C      285    end;
473:C      286    end;
474:C      287    end;
475:C      288    end;
476:C      289    end;
477:C      290    end;
478:C      291    end;
479:C      292    end;
480:C      293    end;
481:C      294    end;
482:C      295    end;
483:C      296    end;
484:C      297    end;
485:C      298    end;
486:C      299    end;
487:C      300    end;
488:C      301    end;
489:C      302    end;
490:C      303    end;
491:C      304    end;
492:C      305    end;
493:C      306    end;
494:C      307    end;
495:C      308    end;
496:C      309    end;
497:C      310    end;
498:C      311    end;
499:C      312    end;
500:C      313    end;
501:C      314    end;
502:C      315    end;
503:C      316    end;
504:C      317    end;
505:C      318    end;
506:C      319    end;
507:C      320    end;
508:C      321    end;
509:C      322    end;
510:C      323    end;
511:C      324    end;
512:C      325    end;
513:C      326    end;
514:C      327    end;
515:C      328    end;
516:C      329    end;
517:C      330    end;
518:C      331    end;
519:C      332    end;
520:C      333    end;
521:C      334    end;
522:C      335    end;
523:C      336    end;
524:C      337    end;
525:C      338    end;
526:C      339    end;
527:C      340    end;
528:C      341    end;
529:C      342    end;
530:C      343    end;
531:C      344    end;
532:C      345    end;
533:C      346    end;
534:C      347    end;
535:C      348    end;
536:C      349    end;
537:C      350    end;
538:C      351    end;
539:C      352    end;
540:C      353    end;
541:C      354    end;
542:C      355    end;
543:C      356    end;
544:C      357    end;
545:C      358    end;
546:C      359    end;
547:C      360    end;
548:C      361    end;
549:C      362    end;
550:C      363    end;
551:C      364    end;
552:C      365    end;
553:C      366    end;
554:C      367    end;
555:C      368    end;
556:C      369    end;
557:C      370    end;
558:C      371    end;
559:C      372    end;
560:C      373    end;
561:C      374    end;
562:C      375    end;
563:C      376    end;
564:C      377    end;
565:C      378    end;
566:C      379    end;
567:C      380    end;
568:C      381    end;
569:C      382    end;
570:C      383    end;
571:C      384    end;
572:C      385    end;
573:C      386    end;
574:C      387    end;
575:C      388    end;
576:C      389    end;
577:C      390    end;
578:C      391    end;
579:C      392    end;
580:C      393    end;
581:C      394    end;
582:C      395    end;
583:C      396    end;
584:C      397    end;
585:C      398    end;
586:C      399    end;
587:C      400    end;
588:C      401    end;
589:C      402    end;
590:C      403    end;
591:C      404    end;
592:C      405    end;
593:C      406    end;
594:C      407    end;
595:C      408    end;
596:C      409    end;
597:C      410    end;
598:C      411    end;
599:C      412    end;
600:C      413    end;
601:C      414    end;
602:C      415    end;
603:C      416    end;
604:C      417    end;
605:C      418    end;
606:C      419    end;
607:C      420    end;
608:C      421    end;
609:C      422    end;
610:C      423    end;
611:C      424    end;
612:C      425    end;
613:C      426    end;
614:C      427    end;
615:C      428    end;
616:C      429    end;
617:C      430    end;
618:C      431    end;
619:C      432    end;
620:C      433    end;
621:C      434    end;
622:C      435    end;
623:C      436    end;
624:C      437    end;
625:C      438    end;
626:C      439    end;
627:C      440    end;
628:C      441    end;
629:C      442    end;
630:C      443    end;
631:C      444    end;
632:C      445    end;
633:C      446    end;
634:C      447    end;
635:C      448    end;
636:C      449    end;
637:C      450    end;
638:C      451    end;
639:C      452    end;
640:C      453    end;
641:C      454    end;
642:C      455    end;
643:C      456    end;
644:C      457    end;
645:C      458    end;
646:C      459    end;
647:C      460    end;
648:C      461    end;
649:C      462    end;
650:C      463    end;
651:C      464    end;
652:C      465    end;
653:C      466    end;
654:C      467    end;
655:C      468    end;
656:C      469    end;
657:C      470    end;
658:C      471    end;
659:C      472    end;
660:C      473    end;
661:C      474    end;
662:C      475    end;
663:C      476    end;
664:C      477    end;
665:C      478    end;
666:C      479    end;
667:C      480    end;
668:C      481    end;
669:C      482    end;
670:C      483    end;
671:C      484    end;
672:C      485    end;
673:C      486    end;
674:C      487    end;
675:C      488    end;
676:C      489    end;
677:C      490    end;
678:C      491    end;
679:C      492    end;
680:C      493    end;
681:C      494    end;
682:C      495    end;
683:C      496    end;
684:C      497    end;
685:C      498    end;
686:C      499    end;
687:C      500    end;
688:C      501    end;
689:C      502    end;
690:C      503    end;
691:C      504    end;
692:C      505    end;
693:C      506    end;
694:C      507    end;
695:C      508    end;
696:C      509    end;
697:C      510    end;
698:C      511    end;
699:C      512    end;
700:C      513    end;
701:C      514    end;
702:C      515    end;
703:C      516    end;
704:C      517    end;
705:C      518    end;
706:C      519    end;
707:C      520    end;
708:C      521    end;
709:C      522    end;
710:C      523    end;
711:C      524    end;
712:C      525    end;
713:C      526    end;
714:C      527    end;
715:C      528    end;
716:C      529    end;
717:C      530    end;
718:C      531    end;
719:C      532    end;
720:C      533    end;
721:C      534    end;
722:C      535    end;
723:C      536    end;
724:C      537    end;
725:C      538    end;
726:C      539    end;
727:C      540    end;
728:C      541    end;
729:C      542    end;
730:C      543    end;
731:C      544    end;
732:C      545    end;
733:C      546    end;
734:C      547    end;
735:C      548    end;
736:C      549    end;
737:C      550    end;
738:C      551    end;
739:C      552    end;
740:C      553    end;
741:C      554    end;
742:C      555    end;
743:C      556    end;
744:C      557    end;
745:C      558    end;
746:C      559    end;
747:C      560    end;
748:C      561    end;
749:C      562    end;
750:C      563    end;
751:C      564    end;
752:C      565    end;
753:C      566    end;
754:C      567    end;
755:C      568    end;
756:C      569    end;
757:C      570    end;
758:C      571    end;
759:C      572    end;
760:C      573    end;
761:C      574    end;
762:C      575    end;
763:C      576    end;
764:C      577    end;
765:C      578    end;
766:C      579    end;
767:C      580    end;
768:C      581    end;
769:C      582    end;
770:C      583    end;
771:C      584    end;
772:C      585    end;
773:C      586    end;
774:C      587    end;
775:C      588    end;
776:C      589    end;
777:C      590    end;
778:C      591    end;
779:C      592    end;
780:C      593    end;
781:C      594    end;
782:C      595    end;
783:C      596    end;
784:C      597    end;
785:C      598    end;
786:C      599    end;
787:C      600    end;
788:C      601    end;
789:C      602    end;
790:C      603    end;
791:C      604    end;
792:C      605    end;
793:C      606    end;
794:C      607    end;
795:C      608    end;
796:C      609    end;
797:C      610    end;
798:C      611    end;
799:C      612    end;
800:C      613    end;
801:C      614    end;
802:C      615    end;
803:C      616    end;
804:C      617    end;
805:C      618    end;
806:C      619    end;
807:C      620    end;
808:C      621    end;
809:C      622    end;
810:C      623    end;
811:C      624    end;
812:C      625    end;
813:C      626    end;
814:C      627    end;
815:C      628    end;
816:C      629    end;
817:C      630    end;
818:C      631    end;
819:C      632    end;
820:C      633    end;
821:C      634    end;
822:C      635    end;
823:C      636    end;
824:C      637    end;
825:C      638    end;
826:C      639    end;
827:C      640    end;
828:C      641    end;
829:C      642    end;
830:C      643    end;
831:C      644    end;
832:C      645    end;
833:C      646    end;
834:C      647    end;
835:C      648    end;
836:C      649    end;
837:C      650    end;
838:C      651    end;
839:C      652    end;
840:C      653    end;
841:C      654    end;
842:C      655    end;
843:C      656    end;
844:C      657    end;
845:C      658    end;
846:C      659    end;
847:C      660    end;
848:C      661    end;
849:C      662    end;
850:C      663    end;
851:C      664    end;
852:C      665    end;
853:C      666    end;
854:C      667    end;
855:C      668    end;
856:C      669    end;
857:C      670    end;
858:C      671    end;
859:C      672    end;
860:C      673    end;
861:C      674    end;
862:C      675    end;
863:C      676    end;
864:C      677    end;
865:C      678    end;
866:C      679    end;
867:C      680    end;
868:C      681    end;
869:C      682    end;
870:C      683    end;
871:C      684    end;
872:C      685    end;
873:C      686    end;
874:C      687    end;
875:C      688    end;
876:C      689    end;
877:C      690    end;
878:C      691    end;
879:C      692    end;
880:C      693    end;
881:C      694    end;
882:C      695    end;
883:C      696    end;
884:C      697    end;
885:C      698    end;
886:C      699    end;
887:C      700    end;
888:C      701    end;
889:C      702    end;
890:C      703    end;
891:C      704    end;
892:C      705    end;
893:C      706    end;
894:C      707    end;
895:C      708    end;
896:C      709    end;
897:C      710    end;
898:C      711    end;
899:C      712    end;
900:C      713    end;
901:C      714    end;
902:C      715    end;
903:C      716    end;
904:C      717    end;
905:C      718    end;
906:C      719    end;
907:C      720    end;
908:C      721    end;
909:C      722    end;
910:C      723    end;
911:C      724    end;
912:C      725    end;
913:C      726    end;
914:C      727    end;
915:C      728    end;
916:C      729    end;
917:C      730    end;
918:C      731    end;
919:C      732    end;
920:C      733    end;
921:C      734    end;
922:C      735    end;
923:C      736    end;
924:C      737    end;
925:C      738    end;
926:C      739    end;
927:C      740    end;
928:C      741    end;
929:C      742    end;
930:C      743    end;
931:C      744    end;
932:C      745    end;
933:C      746    end;
934:C      747    end;
935:C      748    end;
936:C      749    end;
937:C      750    end;
938:C      751    end;
939:C      752    end;
940:C      753    end;
941:C      754    end;
942:C      755    end;
943:C      756    end;
944:C      757    end;
945:C      758    end;
946:C      759    end;
947:C      760    end;
948:C      761    end;
949:C      762    end;
950:C      763    end;
951:C      764    end;
952:C      765    end;
953:C      766    end;
954:C      767    end;
955:C      768    end;
956:C      769    end;
957:C      770    end;
958:C      771    end;
959:C      772    end;
960:C      773    end;
961:C      774    end;
962:C      775    end;
963:C      776    end;
964:C      777    end;
965:C      778    end;
966:C      779    end;
967:C      780    end;
968:C      781    end;
969:C      782    end;
970:C      783    end;
971:C      784    end;
972:C      785    end;
973:C      786    end;
974:C      787    end;
975:C      788    end;
976:C      789    end;
977:C      790    end;
978:C      791    end;
979:C      792    end;
980:C      793    end;
981:C      794    end;
982:C      795    end;
983:C      796    end;
984:C      797    end;
985:C      798    end;
986:C      799    end;
987:C      800    end;
988:C      801    end;
989:C      802    end;
990:C      803    end;
991:C      804    end;
992:C      805    end;
993:C      806    end;
994:C      807    end;
995:C      808    end;
996:C      809    end;
997:C      810    end;
998:C      811    end;
999:C      812    end;
1000:C      813    end;

```



```

240:D 2 'OPKLQUERTYUIASDFGHJMKZCVBN']. (shifted)
241:D 2 js = aspecialtype['0,+123+456*789/E({)~1234567890={}; (shifted)
242:D 2 '0,+123+456*789/'\~!@#$%^&*()_+{}~<?>'; (shifted)
243:D 2 omanimalspecials = aspecialtype[
244:D 2 '0,+123+456*789/E({)~!184#64#35#247#248'^([9'~246#254'3]/'#96'<> ',
245:D 2 '0,+123+456*789/'\~!184#64#35#247#248'^({9'~176#254'3]/'#39'<> '};
246:D 2 { HMOVE LINE MODIFIED TO GIVE SHIFT-EXTEND 6 AS ^, NOT I 5/7/84 SFB }
247:D 2 omanalphabet = bytedalphabettype
248:D 2 [b26[214,241,191,187,000,126,215,168,169,170,171,172,212,
249:D 2 222,228,190,186,188,036,250,000,236,181,189,252,249],
250:D 2 b26[210,240,191,187,000,126,211,168,169,170,171,172,208,
251:D 2 222,227,190,186,188,036,250,000,235,180,189,252,249]];
252:D 2
253:D 2 va'
254:D 2 krec : shortint;
-10 255:D 2 krec : langkeyrec;
256:C 2 begin
257:C 2 if langtable[0]=nil then new(langtable[0]);
258:C 2 with langtable[0]^do
259:C 2 begin
260:C 2 langcode := us_kbd;
261:C 2 semantics := generaltrans; { default translator }
262:C 2 can_noadv := true;
263:C 2 kbdcapslock := true;
264:C 2 kbdaltlock := false;
265:C 2 transmode:=kshift_extc;
266:C 2 krec.no_capslock:=true;
267:C 2 krec.no_shift :=false;
268:C 2 krec.no_control:=false;
269:C 2 krec.no_extension:=false;
270:C 2
271:C 2 krec.keyclass :=alpha_key;
272:C 2 { key codes 1..3 }
273:C 2 krec.keys[false]:='~'; krec.keys[true]:='~';
274:C 2 keytable[1]:=krec;
275:C 2 krec.keys[false]:='|'; krec.keys[true]:='|';
276:C 2 keytable[2]:=krec;
277:C 2 krec.keys[false]:=#27; krec.keys[true]:=#127; {esc, del}
278:C 2 keytable[3]:=krec;
279:C 2 keytable[3].no_extension:=true; keytable[2].no_control:=true;
280:C 2
281:C 2 for i:=60 to 125 do
282:C 2 begin
283:C 2 if i>=100 then
284:C 2 begin { normal alpha keys }
285:C 2 if i=100 then krec.no_capslock:=false;
286:C 2 krec.keys[false]:=alphabet[false,i]; krec.keys[true]:=alphabet[true,i];
287:C 2 end
288:C 2 else
289:C 2 begin { special alpha keys }
290:C 2 krec.keys[false]:=us[false,i]; krec.keys[true]:=us[true,i];
291:C 2 end;
292:C 2 keytable[i]:=krec;
293:C 2 end;
294:C 2
295:C 2 { un_implemented key codes }
296:C 2 keytable[0].keyclass:=ignored_key; keytable[4].keyclass:=ignored_key;
297:C 2 keytable[126].keyclass:=ignored_key; keytable[127].keyclass:=ignored_key;
298:C 2
299:C 2 krec.keyclass:=special_key;
300:C 2 krec.no_shift:=false; krec.no_control:=false;

```

```

300:C 3 krec.no_capslock:=true; krec.no_extension:=true;
301:C 3 for i:=5 to 59 do
302:C 4 begin
303:C 4 { special function keys }
304:C 4 krec.keys[false]:=chr(i); krec.keys[true]:=chr(i);
305:C 4 keytable[i]:=krec;
306:C 4 end;
307:C 3 { rightchar{fsp} involved in non advancing keys }
308:C 3 keytable[39].keys[false]:=rightchar; keytable[39].keys[true]:=rightchar;
309:C 3 end; { with langtable[0] }
310:C 2 if kbdtype=itfkbd then
311:C 3 begin
312:C 3 if langtable[1]=nil then new(langtable[1]);
313:C 3 langtable[1]^:=langtable[0]^; { copy US to alternate }
314:C 3 with langtable[1] do
315:C 4 begin
316:C 4 { load roman keyboard }
317:C 4 langcode:=roman8_kbd;
318:C 4 keytable[1].keys[false]:=#251; keytable[1].keys[true]:=#253;(<<,>>)
319:C 4 for i:=60 to 125 do
320:C 5 begin
321:C 5 if i>=80 then keytable[i].no_capslock:=true;
322:C 5 with keytable[i] do
323:C 6 if i>=100 then
324:C 7 begin
325:C 7 { normal alpha keys }
326:C 7 keys[false]:=chr(romanalphabet[false,i]);
327:C 7 keys[true]:= chr(romanalphabet[true,i]);
328:C 7 end
329:C 7 else
330:C 7 begin { special alpha keys }
331:C 7 keys[false]:=romanspecials[false,i];
332:C 7 keys[true]:= romanspecials[true,i];
333:C 7 end;
334:C 5 end;
335:C 4 { mark non advancing keys }
336:C 4 for i:=107 to 111 do keytable[i].keyclass:=nonadv_key;
337:C 4 { ignore some keys }
338:C 4 keytable[102].keyclass:=ignored_key;
339:C 4 keytable[104].keyclass:=ignored_key;
340:C 4 keytable[120].keyclass:=ignored_key;
341:C 4 { cancel extension interpretation on all keys }
342:C 4 for i:=0 to 127 do keytable[i].no_extension:=true;
343:C 4 end; { with langtable[1] }
344:C 3 end; { if ITF keyboard }
345:C 2 end; { initlang }
346:C 1 procedure clearanychar;
347:C 2 begin
348:C 2 if anychar then
349:C 3 begin anychar := false; kbdisrhook := anycharsavehook; end;
350:C 2 end; { clearanychar }
351:C 1 procedure rpghandler(var kbdstatus, kbddata: byte; var dokey: boolean);
352:C 2 var key: char;
353:C 2 begin
354:C 2 { only store the key if the typeahead buffer is empty }
355:C 2 if kbdstatus div 16 of
356:C 3 14: { shifted }

```



```

360:C      3      if kbddata >= 128 then key:=chr(lf) else key:=chr(us);
361:C      3      15: (unshifted)
362:C      3      if kbddata >= 128 then key:=chr(fsp) else key:=chr(bs);
363:C      3      otherwise ( ignore if control key down )
364:C      3      beep; key := ' ';
365:C      3      end;
366:C      2      if (keybuffer^.size=0) and (key<>' ') then
367:C      3      begin
368:C      3          keybufops(kappend,key); clearanychar;
369:C      3          call(kbdreleasehook); { signal non empty buffer }
370:C      3      end;
371:C      2      end; { rpghandler }
372:S
373:D      1      procedure keyservice(var kbdstatus, kbddata: byte; var dokey: boolean);
374:D      2      label 1;
375:D      2      var done: boolean;
376:D      -2      small: boolean;
377:D      -4      i: shortint;
378:D      -5      c: char;
379:S
380:D      2      procedure morealpha;
381:C      3      begin
382:C      3          done := true;
383:C      3          if not alphastate then call(togglealphahook)
384:C      3          else if graphicstate then call(togglegraphicshook);
385:C      3      end;
386:D      -5      2
387:D      2      procedure moregraphics;
388:C      3      begin
389:C      3          done := true;
390:C      3          if not graphicstate then call(togglegraphicshook)
391:C      3          else if alphastate then call(togglealphahook);
392:C      3      end;
393:D      -5      2
394:D      2      procedure dumpalpha;
395:C      3      begin done := true; lockedaction(dumpalphahook);
396:C      3      end;
397:D      -5      2
398:D      2      procedure dumpgraphics;
399:C      3      begin done := true; lockedaction(dumpgraphicshook);
400:C      3      end;
401:D      -5      2
402:D      2      procedure unrecognized;
403:C      3      begin done := true; beep;
404:C      3      end;
405:S
406:D      2      procedure remove(all:boolean);
407:D      -1      3      var dummykey: char;
408:C      3      begin
409:C      3          if keybuffer^.size>0 then
410:C      4          begin
411:C      4              if all then keybufops(kclear,dummykey)
412:C      5              else keybufops(kgetlast,dummykey); { pop last char }
413:C      4          end
414:C      4          else beep;
415:C      3          done:=true;
416:C      3          end; { remove }
417:S
418:D      2      procedure debugkey;
419:C      3      begin call(debugger,3,langcom.status,langcom.data); end;

```

```

420:S      2      begin { keyservice }
421:C      2      if dokey then
422:C      3      with langcom do
423:C      4      begin
424:C      4          done := false; {done indicates that key is handled immediately}
425:C      4          status := kbdstatus;
426:C      4          data := kbddata;
427:C      4          extension:= not odd(kbdstatus div 8);
428:C      4          shift := not odd(kbdstatus div 16);
429:C      4          control := not odd(kbdstatus div 32);
430:C      4
431:S
432:C      4          if not odd(status) then
433:C      5          begin result:=alpha_key; key:=chr(data); end
434:C      5          else call(langtable[langindex]^semantics);
435:S
436:C      4          case result of
437:C      5              alpha_key, { don't do anything yet }
438:C      5              nonadv_key, { have non advancing key }
439:C      5              nona_alpha_key: (* 3.01 BUG FIX 9/19/84 SFB-JWS *);
440:C      5              special_key:
441:C      5              begin { decode special function keys }
442:C      5                  small := kbdtype=smallkbd;
443:C      5                  case data of
444:C      6                      17: if control and not shift then unrecognized {enter/print}
445:C      7                      else if shift and control then dumpgraphics
446:C      8                      else if shift then dumpalpha
447:C      9                      else key:=chr(cr); {enter}
448:C      6
449:C      6                  20:begin {system/user}
450:C      6                      if shift then key:='U' else key:='5';
451:C      6                      kbdsysmode:=not shift;
452:C      6                      setstatus(6,key);
453:C      7                      if key='U' then
454:C      8                      if (menustate=m_sysnorm) or (menustate=m_sysshift) then
455:C      8                      begin
456:C      8                          menustate:=m_none;
457:C      8                          keybuffer^.echo:=true;
458:C      8                          keybufops(kdisplay,c);
459:C      8                      end;
460:C      6                      done:=true;
461:C      6                  21:begin {menu}
462:C      6                      if kbdsysmode then
463:C      7                      begin
464:C      7                          call(crtllhook,cliclear,i,c); { clear display }
465:C      7                          if menustate=m_sysshift then menustate:=m_none;
466:C      7                          menustate:=mstates[menustate,shift];
467:C      7                          keybuffer^.echo:=(menustate=m_none);
468:C      7                          case menustate of
469:C      8                              m_none : keybufops(kdisplay,c);
470:C      8                              m_sysnorm : call(crtllhook,cldisplay,sysmenu^,c);
471:C      8                              m_sysshift: call(crtllhook,cldisplay,sysmenushift^,c);
472:C      8                              otherwise
473:C      8                                  end;
474:C      7                          done := true;
475:C      7                      end
476:C      6                      else unrecognized;
477:C      6                      end;
478:C      6                  22:if control then remove(true) else key:=chr(del); { clear line }
479:C      6                  23:if control then remove(true) else key:=chr(ff); { clear display }

```



```

480:C      6      24:begin KBDCAPSLOCK := not KBDCAPSLOCK; done := true end;
481:C      6      9,25:      key := chr(tab);      (tab)
482:C      6      34:      key := chr(lf);      (down arrow)
483:C      6      35:      key := chr(us);      (up arrow)
484:C      6      38,46: if control then remove(false) else key := chr(bs);      (left arrow, backspace)
485:C      6      39:BEGIN (SFB 5/21/84) key := chr(fsp); STATUS:=0;      (right arrow)
486:C      6      CALL (LANGTABLE[LANGINDEX]^SEMANTICS);END;
487:C      6      40,43:      key := 'I';      (insert mode)
488:C      6      41: if small then moregraphics else
489:C      6      42: if shift and small then morealpha else unrecognized;
490:C      6      44:      key := 'D';      (delete mode)
491:C      6      47:      key := 'R';      (recall)
492:C      6      48:      key := 'E';      (delete mode)
493:C      6      49: if shift then dumpalpha else morealpha;
494:C      6      50: if shift then dumpgraphics else moregraphics;
495:C      6      51: begin      (STEP) (ANYCHAR)
496:C      6      done := true;
497:C      6      if shift then begin anychar:=true;
498:C      6      anycharsavehook := kbdisrhook;
499:C      6      kbdisrhook := keyservice;
500:C      6      buildcount := 1; buildchar:= 0;
501:C      6      end
502:C      6      else debugkey;      (STEP key)
503:C      6      end;
504:C      6      52: if control then remove(true)
505:C      6      else if shift then key := chr(ff);      (clear screen)
506:C      6      53: if shift and small then dumpalpha else unrecognized;
507:C      6      54: if shift and small then dumpgraphics else unrecognized;
508:C      6      6,55: begin      (result, set tab)
509:C      6      done := true;
510:C      6      clearanychar; lockedaction(stopaction);
511:C      6      end;
512:C      6      5,56: begin      (pause)
513:C      6      done := true; data:=56;      (change 5 to 56)
514:C      6      if locklevel = 0 then debugkey
515:C      6      else if control then lockedaction(cntlrpausekey)
516:C      6      else lockedaction(pausekey);
517:C      6      end;
518:C      6      8,57:      key := chr(cr);      (ENTER)
519:C      6      58: begin done := true; debugkey; end;
520:C      6      7,59:      (EXECUTE key)
521:C      6      if control then key := chr(ctrl)
522:C      6      else if shift then key := chr(esc)
523:C      6      else key := chr(etx);
524:C      6      otherwise unrecognized; (no such code)
525:C      6      end; (decode special function keys)
526:C      6      ignored key; goto 1; (ignore this key)
527:C      6      end; (case langcom.result)
528:C      5
529:C      5
530:C      5
531:C      5
532:C      5
533:C      5
534:C      5
535:C      5
536:C      4
537:C      5
538:C      5
539:C      6

```

```

540:C      6      if not (shift and (data = 51)) then clearanychar;
541:C      6      end
542:C      6      else
543:C      6      begin
544:C      6      if {key < '0'} or {key > '9'} then
545:C      6      begin clearanychar; unrecognized; end
546:C      6      else
547:C      6      begin
548:C      6      buildchar:=buildchar*10+(ord(key)-ORD('0'));
549:C      6      buildcount:=buildcount+1;
550:C      6      done := buildcount <= 3;
551:C      6      if not done then
552:C      6      begin
553:C      6      clearanychar; key:=chr(buildchar mod 256);
554:C      6      status:=0;
555:C      6      call(langtable[langindex]^semantics);
556:C      6      end;
557:C      6      end;
558:C      6      end; (anychar)
559:C      5
560:C      5
561:C      4
562:C      5
563:C      5
564:C      6
565:C      6
566:C      7
567:C      7
568:C      7
569:C      7
570:C      4
571:C      4
572:C      2
573:C      2
574:C      1
575:C      2
576:C      2
577:C      1
578:C      2
579:C      2
580:C      2
581:C      2
582:C      2
583:C      2
584:C      2
585:C      2
586:C      2
587:C      2
588:C      2
589:C      2
590:C      2
591:C      2
592:C      1
593:C      1
594:C      1
595:C      1
596:C      1
597:C      1

```

***WARNING: (line 584): 'ADDR' of a constant may not be supported on other implementations
 ***WARNING: (line 585): 'ADDR' of a constant may not be supported on other implementations
 ***WARNING: (line 586): 'ADDR' of a constant may not be supported on other implementations

```

584:C      2      sysmenu := addr(synorm);
585:C      2      sysmushift := addr(syshft);
586:C      2      initlang;
587:C      2      KBDSETUP(SET_AUTO_REPEAT,4);      (auto repeat period = 40 ms)
588:C      2      KBDSETUP(SET_AUTO_DELAY,30);      (auto repeat delay = 300 ms)
589:C      2      SETPRGRATE(1);      (rpg interrupt rate = 10 ms)
590:C      2      end; (initkeys)
591:C      2
592:C      1
593:C      1
594:C      1
595:C      1
596:C      1
597:C      1

```



```
598:C      1  initkeys; markuser;  
599:C      1  end.  
600:S
```

No errors. 3 warnings.

***** Nonstandard language features enabled *****

Description

LIB provides the normal user-interface-level routines for DGL.

Requirements

DGL_TYPES, SYSGLOBALS, SYSDEVS, DGL_VARS, DGL_AUTL, DGL_GEN, ASM, GLE_GEN, GLE_TYPES, GLE_GENI, DGL_CONFIG_OUT, DGL_CONFIG_IN, DGL_TOOLS, DGL_IBODY, and IODECLARATIONS.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:D 0 {
2:D 0 { Pascal work station graphics library }
3:D 0 {
4:D 0 { Module = DGL_LIB }
5:D 0 { Programmer = BJS }
6:D 0 { Date = 2/1/81 }
7:D 0 {
8:D 0 { Rev history:
9:D 0 { 5/21/82 BJS Set display and locator names to ' ' on term
10:D 0 { 5/21/82 BJS Fixed inverted window/set_echo_pos bug
11:D 0 { 8/25/82 BJS Major mods for GLE
12:D 0 { 2/17/84 BDS Changed dynamic to global allocation for Pascal 3.0
13:D 0 { Purpose: Hold normal user interface routines
14:D 0 {
15:D 0 { (c) Copyright Hewlett-Packard Company, 1983.
16:D 0 { All rights are reserved. Copying or other
17:D 0 { reproduction of this program except for archival
18:D 0 { purposes is prohibited without the prior
19:D 0 { written consent of Hewlett-Packard Company.
20:D 0 {
21:D 0 {
22:D 0 { RESTRICTED RIGHTS LEGEND
23:D 0 {
24:D 0 { Use, duplication, or disclosure by the Government
25:D 0 { is subject to restrictions as set forth in
26:D 0 { paragraph (b) (3) (B) of the Rights in Technical
27:D 0 { Data and Computer Software clause in
28:D 0 { DAR 7-104.9(a).
29:D 0 {
30:D 0 { HEWLETT-PACKARD COMPANY
31:D 0 { Fort Collins, Colorado }
32:D 0 {
33:D 0 $mod:$
34:D 0 $include 'OPTIONS'$
35:D 0 { This include file specifies range checking, debug and other compiler
36:D 0 { options for the graphics library }
37:D 0 {
38:D 0 $debug OFF$
39:D 0 $range OFF$
40:D 0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
41:D 0 $FLORT_HDW TEST$
42:D 0 {
43:D 0 {
44:D 0 {
45:D 0 {
46:D 0 {
20000:D 0 $include 'OPTIONS'$
20001:D 0 $linenum 20000$
20002:D 0 $search 'TYPES'
20003:D 0 {
20004:D 0 { 'DGL_VARS',
20005:D 0 { 'GEN'
20006:D 0 { 'GLE_IB',
20007:D 0 { 'DGL_C_OUT',
20008:D 0 { 'DGL_C_IN',
20009:D 0 { 'DGL_TOOL',
20010:D 0 { 'DGL_AUTL',
20011:D 0 { 'DGL_IBODY'$
20012:D 0 module DGL_LIB;

```

```

20013:S 1 import dgl_types;
20014:D 1
20015:S 1 export
20016:D 1
20017:D 1 procedure graphics_init;
20018:D 1 procedure set_aspect (x, y : real);
20019:D 1 procedure set_viewport (vxmin, vxmax, vymin, vmax : real);
20020:D 1 procedure set_window (wxmin, wxmax, wymin, wymax : real);
20021:D 1 procedure display_finit ( fname : gstring255;
20022:D 2 device_name : gstring255;
20023:D 2 control : integer;
20024:D 2 var ierr : integer);
20025:D 1 procedure display_init ( dev_adr : integer;
20026:D 2 control : integer;
20027:D 2 var ierr : integer);
20028:D 1 procedure display_term;
20029:D 1 procedure make_pic_current;
20030:D 1 procedure set_timing (opcode : integer);
20031:D 1 procedure graphics_term;
20032:D 1 procedure locator_term;
20033:S 1
20034:D 1 procedure set_color_model (model : integer);
20035:D 1 procedure set_line_style (index : integer);
20036:D 1 procedure set_color (index : integer);
20037:D 1 procedure set_line_width (index : integer);
20038:D 1 procedure locator_init ( dev_adr : integer;
20039:D 2 var ierr : integer);
20040:S 1
20041:D 1 function graphicerror : integer;
20042:S 1
20043:D 1 procedure move (x,y:real);
20044:D 1 procedure line (x,y:real);
20045:D 1 procedure int_polyline ( num_points : integer;
20046:D 2 anyvar xvec, yvec : gshortint_list );
20047:D 1 procedure polyline ( num_points : integer;
20048:D 2 anyvar xvec, yvec : greal_list );
20049:D 1 procedure int_move (ix,iy : gshortint);
20050:D 1 procedure int_line (ix,iy : gshortint);
20051:D 1 procedure marker (marker_number : integer);
20052:D 1 procedure set_display_lim ( dxmin, dxmax, dymin, dymax : real;
20053:D 2 var ierr : integer);
20054:D 1
20055:D 1 procedure clear_display;
20056:D 1 procedure input_esc ( opcode : integer;
20057:D 2 isize : integer;
20058:D 2 rsize : integer;
20059:D 2 anyvar llist : gint_list;
20060:D 2 anyvar rlist : greal_list;
20061:D 2 var ierr : integer );
20062:D 1
20063:D 1 procedure output_esc ( opcode : integer;
20064:D 2 isize : integer;
20065:D 2 rsize : integer;
20066:D 2 anyvar llist : gint_list;
20067:D 2 anyvar rlist : greal_list;
20068:D 2 var ierr : integer );
20069:S 1
20070:D 1 procedure set_text_rot (dx, dy : real);
20071:D 1 procedure set_char_size (width,height : real);
20072:D 1 procedure gtext( s : gstring255 );

```



```

20073:S
20074:D 1 procedure set_echo_pos (wx,wy : real);
20075:S
20076:D 1 procedure await_locator ( echo : integer;
20077:D 2 var button : integer;
20078:D 2 var rx, ry : real );
20079:S
20080:D 1 procedure sample_locator ( echo : integer;
20081:D 2 var rx, ry : real );
20082:S
20083:D 1 procedure set_locator_lim ( lxmin,lxmax,lymin,lymax : real;
20084:D 2 var ierr : integer );
20085:S
20086:D 1 procedure set_color_table ( index : integer;
20087:D 2 parm1 : real;
20088:D 2 parm2 : real;
20089:D 2 parm3 : real);
20090:S
20091:D 1 procedure convert_wtodmm ( wx, wy : real; var mmx,mmy : real);
20092:D 1 procedure convert_wtoimmm ( wx, wy : real; var mmx,mmy : real);
20093:S
20094:D 1 implement
20095:S
20096:D 1 import sysglobals,
20097:D 1 sysdevs,
20098:S {kbd,
20099:D 1 keys,}
20100:D 1 {asm,}
20101:D 1 dgl_vars,
20102:D 1 dgl_autl,
20103:D 1 dgl_gen,
20104:D 1 gle_types,
20105:D 1 gle_GEN,
20106:D 1 gle_genl,
20107:D 1 dgl_config_out,
20108:D 1 dgl_config_in,
20109:D 1 dgl_tools,
20110:D 1 dgl_ibody,
20111:D 1 ideclarations;
20112:S
20113:D 1 {procedure hpm_dispose ( var object : anyptr; bytesize : integer ); external;}
20114:S
20115:D 1 function graphicerror : integer;
20116:S
20117:D 2 { Purpose: To return the most resent graphics error number }
20118:S
20119:C 2 begin
20120:C 2 graphicerror := graphics_error;
20121:C 2 end; { graphicerror }
20122:S
20123:D 1 procedure display_move ( x,y : integer );
20124:S
20125:C 2 begin
20126:C 2 with gle_gcb^ do
20127:C 3 begin
20128:C 3 end_x := x;
20129:C 3 end_y := y;
20130:C 3 call (move,gle_gcb);
20131:C 3 end;
20132:C 2 end;

```

```

20133:S
20134:D 1 procedure display_draw ( x,y : integer );
20135:S
20136:C 2 begin
20137:C 2 with gle_gcb^ do
20138:C 3 begin
20139:C 3 end_x := x;
20140:C 3 end_y := y;
20141:C 3 call (draw,gle_gcb);
20142:C 3 end;
20143:C 2 end;
20144:S
20145:D 1 procedure adjust_echo (var dx,dy : integer);
20146:S
20147:D 2 { Purpose : To adjust echo for rubber band line effects }
20148:S
20149:C 2 begin
20150:C 2 case current_echo_type of
20151:C 3 5 : dy:=d_loc_echo_y; { horz rubber band line }
20152:C 3 6 : dx:=d_loc_echo_x; { vert rubber band line }
20153:C 3 7 : { snap horz / vert rubber band line }
20154:C 3 if abs(dx-d_loc_echo_x) >= abs(dy-d_loc_echo_y) then
20155:C 4 dy:=d_loc_echo_y
20156:C 4 else
20157:C 4 dx:=d_loc_echo_x;
20158:C 3 otherwise { all other echos are ok }
20159:C 3 end; { of case }
20160:C 2 end; { adjust_echo }
20161:S
20162:D 1 procedure cursor ( x,y : integer);
20163:S
20164:C 2 begin
20165:C 2 display_move(x-8,y);
20166:C 2 display_draw(x+8,y);
20167:S
20168:C 2 display_move(x,y+8);
20169:C 2 display_draw(x,y-8);
20170:C 2 end;
20171:S
20172:D 1 procedure echo_cursor (dx,dy : integer);
20173:S
20174:D 2 { Purpose : To perform the current echo on a raster display }
20175:S
20176:C 2 begin
20177:C 2 case current_echo_type of
20178:C 3 1,2 : cursor(dx,dy);
20179:S
20180:C 3 3 :
20181:C 3 3 : with gcb^gcb^.max_disp_lim do
20182:C 4 begin {full screen}
20183:C 4 display_move(trunc(xmin),dy);
20184:C 4 display_draw(trunc(xmax),dy);
20185:C 4 display_move(dx,trunc(ymmin));
20186:C 4 display_draw(dx,trunc(ymax));
20187:C 4 display_move(dx,dy); { set cp to cursor center }
20188:C 4 end;
20189:C 3 4,5,6,7 :
20190:C 3
20191:C 3
20192:C 3

```



```

20193:C      3      begin (rubber bands)
20194:C      3      adjust_echo(dx,dy);      { are dx, dy correct for this echo? }
20195:C      3      display_move(d_loc_echo_x,d_loc_echo_y);
20196:C      3      display_draw(dx,dy);
20197:C      3      cursor(dx,dy);
20198:C      3      end;
20199:S
20200:C      3      8 :
20201:C      3      begin (rubber band box)
20202:C      3      display_move(d_loc_echo_x,d_loc_echo_y);
20203:C      3      display_draw(d_loc_echo_x,dy);
20204:C      3      display_draw(dx,dy);
20205:C      3      display_draw(dx,d_loc_echo_y);
20206:C      3      display_draw(d_loc_echo_x,d_loc_echo_y);
20207:C      3      cursor(dx,dy);
20208:C      3      end;
20209:S
20210:C      3      otherwise ;      { no echo }
20211:S
20212:C      3      end; { of case }
20213:C      2
20214:C      2 end; { display_echo }
20215:S
20216:C      1 procedure DGL_CURSOR ( gle_gcb : graphics_control_block_ptr );
20217:S
20218:D      2 VAR
20219:D      -8 2 x,y : integer;
20220:D      -12 2 CPX : INTEGER;
20221:D      -16 2 CPY : INTEGER;
20222:S
20223:C      2 begin
20224:C      2 with gle_gcb^ do
20225:C      3 begin
20226:C      3 x := info1;
20227:C      3 y := info2;
20228:C      3 CPX := CURRENT_POS_X; { SAVE CP }
20229:C      3 CPY := CURRENT_POS_Y;
20230:C      3 gle_wait_blank(gle_gcb);
20231:C      3 if current_cursor_state = 1 then { remove old cursor }
20232:C      4 echo_cursor (current_cursor_x, current_cursor_y);
20233:C      3 if (info3 = 1) then
20234:C      4 echo_cursor (x,y);      { draw new cursor }
20235:S
20236:C      3 current_cursor_state := info3;
20237:C      3 current_cursor_x := x;
20238:C      3 current_cursor_y := y;
20239:S
20240:C      3 END_X := CPX;      { RESTORE CP }
20241:C      3 END_Y := CPY;
20242:C      3 GLE_MOVE(GLE_GCB);
20243:C      3 end;
20244:C      2 end;
20245:S
20246:D      -16 1 procedure convert_wtodmm ( wx, wy : real; var mmx,mmx : real);
20247:S
20248:D      -16 2 var
20249:D      -32 2 dx,dy : real;
20250:S
20251:C      2 begin
20252:C      2 ck_system_init;

```

```

20253:C      2 ck_display_init;
20254:C      2 dx := wx * xwtod_scale + xwtod_offset;
20255:C      2 dy := wy * ywtod_scale + ywtod_offset;
20256:C      2 with gcb^,gcb^.max_disp_lim,gle_gcb^ do
20257:C      3 begin
20258:C      3 mmx := (dx - xmin) / display_res_x;
20259:C      3 mmy := (dy - ymin) / display_res_y;
20260:C      3 end;
20261:C      2 end;
20262:D      -16 1 procedure convert_wtolmm ( wx, wy : real; var mmx,mmx : real);
20263:S
20264:D      -16 2 var
20265:D      -32 2 dx,dy : real;
20266:D      -48 2 tx,ty : real;
20267:S
20268:C      2 begin
20269:C      2 ck_system_init;
20270:C      2 ck_locator_init;
20271:C      2 dx := wx * xwtod_scale + xwtod_offset;
20272:C      2 dy := wy * ywtod_scale + ywtod_offset;
20273:C      2 with gcb^,gle_gcb^ do
20274:C      3 begin
20275:C      3 { convert display to locator }
20276:C      3 tx := ((dx-cur_disp_lim.xmin) / xltod_scale) + log_loc_lim.xmin;
20277:C      3 ty := ((dy-cur_disp_lim.ymin) / yltod_scale) + log_loc_lim.ymin;
20278:C      3
20279:C      3 { convert to mm }
20280:C      3 mmx := (tx - max_loc_lim.xmin) / input_res_x;
20281:C      3 mmy := (ty - max_loc_lim.ymin) / input_res_y;
20282:C      3 end;
20283:C      2 end;
20284:S
20285:D      -32 1 procedure set_viewport (vxmin, vxmax, vymin, vmax : real );
20286:S
20287:D      -32 2 { Purpose: To set the viewport }
20288:S
20289:C      2 begin
20290:C      2 ck_system_init;
20291:C      2
20292:C      2 { ck perms }
20293:C      2
20294:C      2 if (vxmin >= vxmax) or (vymin >= vmax) then error (err_bad_parms);
20295:C      2
20296:C      2 with gcb^ do
20297:C      3 begin
20298:C      3 if (vxmin < 0.0) or      {ck with vir limits}
20299:C      4 (vymin < 0.0) or
20300:C      4 (vxmax > cur_vir_lim.xlim) or
20301:C      4 (vymax > cur_vir_lim.ylim) then
20302:C      4 error (err_out_virt);
20303:C      4
20304:C      4 with viewport_lim do
20305:C      5 begin
20306:C      5 xmin := vxmin;
20307:C      5 xmax := vxmax;
20308:C      5 ymin := vymin;
20309:C      5 ymax := vmax;
20310:C      5 end;
20311:C      4
20312:S

```



```

20313:C      3      calculate_viewing;
20314:S
20315:C      3      ( set flag so character size will be recalculated )
20316:S
20317:C      3      calc_text_xform := true;
20318:C      3      end
20319:C      3      end; ( set_viewport )
20320:S
20321:D -32 1 procedure set_window (wxmin, wxmax, wymin, ymax : real );
20322:S
20323:D -32 2 ( Purpose: To set the window )
20324:S
20325:C      2      begin
20326:C      2      ck_system_init;
20327:S
20328:C      2      (ck_parms)
20329:S
20330:C      2      if (wxmin = wxmax) or (wymin = ymax) then error (err_bad_parms);
20331:S
20332:C      2      with gcb^ do
20333:C      3      with window_lim do
20334:C      4      begin
20335:C      4      xmin := wxmin;          ( set the new window )
20336:C      4      xmax := wxmax;
20337:C      4      ymin := wymin;
20338:C      4      ymax := ymax;
20339:C      4      end;
20340:S
20341:C      2      calculate_viewing;
20342:S
20343:C      2      ( set flag so character xform will be recalculated )
20344:S
20345:C      2      calc_text_xform := true;
20346:C      2      end; ( set_window )
20347:S
20348:D -16 1 procedure set_aspect (x, y : real);
20349:S
20350:D -16 2 ( Purpose: To set the aspect ratio )
20351:S
20352:C      2      begin
20353:C      2      ck_system_init;
20354:S
20355:C      2      (ck_parms)
20356:C      2      if (x <= 0.0) or (y <= 0.0) then error (err_aspect);
20357:S
20358:C      2      with gcb^ do
20359:C      3      ( calc new limits )
20360:C      3      with cur_vir_lim do
20361:C      4      begin
20362:C      4      aspect_ratio := y / x;
20363:C      4      if aspect_ratio <= 1.0 then
20364:C      5      begin
20365:C      5      xlim := 1.0;
20366:C      5      ylim := aspect_ratio;
20367:C      5      end
20368:C      5      else
20369:C      5      begin
20370:C      5      xlim := 1.0 / aspect_ratio;
20371:C      5      ylim := 1.0;
20372:C      5      end;

```

```

20373:C      4      ( set viewport to new limits )
20374:C      4      set_viewport( 0.0, xlim, 0.0, ylim);
20375:C      4      end;
20376:C      2      end; ( set_aspect )
20377:S
20378:D      1 procedure make_pic_current;
20379:S
20380:C      2      begin
20381:C      2      ck_system_init;
20382:C      2      ck_display_init;
20383:C      2      gle_flush_buffer ( gle_gcb );
20384:C      2      end;
20385:S
20386:D      1 procedure set_timing ( opcode : integer );
20387:S
20388:C      2      begin
20389:C      2      ck_system_init;
20390:C      2      if (opcode < 0) or (opcode > 1) then error(err_bad_parms);
20391:C      2      gcb^.dgl_current_timing_mode := opcode;
20392:C      2      if disp_init then
20393:C      3      begin
20394:C      3      gle_gcb^.info1 := opcode;
20395:C      3      gle_buffer_mode ( gle_gcb );
20396:C      3      end;
20397:C      2      end;
20398:S
20399:D      1 procedure set_color_model ( model : integer);
20400:S
20401:C      2      begin
20402:C      2      ck_system_init;
20403:S
20404:C      2      if (model<1) or (model>2) then error(err_bad_parms);
20405:C      2      gcb^.dgl_current_color_model := model;
20406:C      2      end;
20407:S
20408:D      1 procedure set_color_table ( index : integer;
20409:D      2      parm1 : real;
20410:D      2      parm2 : real;
20411:D      2      parm3 : real);
20412:S
20413:C      2      begin
20414:C      2      ck_system_init;
20415:C      2      ck_display_init;
20416:C      2      with gcb^,gle_gcb^ do
20417:C      3      begin
20418:C      3      if (index >= 0) and (index <= color_table_size) then
20419:C      4      begin
20420:C      4      dgl_polygon_color_current := false; ( dither pattern is wrong )
20421:C      4      if ((0 > parm1) or (parm1 > 1)) or
20422:C      5      ((0 > parm2) or (parm2 > 1)) or
20423:C      5      ((0 > parm3) or (parm3 > 1)) then error (err_bad_parms);
20424:C      4      call (proc_color_table,index,parm1,parm2,parm3);
20425:C      4      { always recalculate line color (2.1 buug fix) }
20426:C      4      call (proc_color,dgl_current_color);
20427:C      4      end;
20428:C      3      end;
20429:C      2      end;
20430:S
20431:D      1 procedure set_line_width (index : integer);
20432:S

```



```

20433:D      2 ( Purpose: To set the line width primitives will be drawn with      )
20434:S      )
20435:C      2 begin
20436:C      2   ck_system_init;
20437:C      2   ck_display_init;
20438:C      2   with gle_gcb^ do
20439:C      3   begin
20440:C      3     if (index < 1) or (index > linewidths) then index := 1;
20441:C      3     gcb^.dgl_current_linewidth := index;
20442:C      3     info1 := index;
20443:C      3     gle_linewidth ( gle_gcb );
20444:C      3   end;
20445:C      2 end; ( set_linewidth )
20446:D      1 procedure set_color (index : integer);
20447:D      1
20448:S      2 ( Purpose: To set the color primitives will be drawn with      )
20449:D      2
20450:S      2 begin
20451:C      2   ck_system_init;
20452:C      2   ck_display_init;
20453:C      2
20454:S      2   with gcb^,gle_gcb^ do
20455:C      3   begin
20456:C      3     if (index < 0) or
20457:C      3       ((index > gamut) and
20458:C      3         ((color_table_size = 0) or (index > color_table_size))) then index := 1;
20459:C      3     ( optimize changing color on raster devices (2.1 bug fix) )
20460:C      3     if ((dgl_current_color <> index) or (complement_support <> 1)) then
20461:C      3       begin
20462:C      3         call (proc_color,index);
20463:C      3         dgl_current_color := index;
20464:C      3       end;
20465:C      3     end;
20466:C      3   end;
20467:C      2 end; ( set_color )
20468:S      1 procedure set_line_style ( index : integer);
20469:D      1
20470:S      2 ( Purpose: To set the linestyle that primitives are drawn with      )
20471:D      2
20472:S      2 begin
20473:C      2   ck_system_init;
20474:C      2   ck_display_init;
20475:C      2
20476:S      2   with gcb^ do
20477:C      3   begin
20478:C      3     if (index < 1) or (index > number_dgl_linestyles) then index := 1;
20479:C      3     (if dgl_current_linestyle <> index then
20480:C      3       begin
20481:C      3         dgl_current_linestyle := index;
20482:C      3         call (proc_linestyle,index);
20483:C      3       end;
20484:C      3     end;
20485:C      3   end;
20486:C      2 end; ( set_line_style )
20487:S      1 procedure set_display_lim ( dxmin, dxmax, dymin, dymax : real;
20488:D      1   var ierr : integer);
20489:D      1
20490:S      -32 2 ( Purpose : To set the logical display limits      )
20491:D      -32 2
20492:S

```

```

20493:D      -32 2 var
20494:D      -40 2   txmin : real;
20495:D      -48 2   txmax : real;
20496:D      -56 2   tymin : real;
20497:D      -64 2   tymax : real;
20498:S
20499:C      2 begin
20500:C      2   ck_system_init;
20501:C      2   ck_display_init;
20502:S
20503:C      2   (WRITELN(dxmin:19:18,dxmax:18:18,dymin:19:18,dymax:19:18));
20504:C      2   { ck parms }
20505:C      2   if ((dxmin < dxmax) and (dymin < dymax)) then
20506:C      3   with gcb^ do
20507:C      4   with max_disp_lim do
20508:C      5   begin
20509:C      5     with gle_gcb^ do
20510:C      6     begin
20511:C      6       txmin := (dxmin * display_res_x) + xmin;
20512:C      6       txmax := (dxmax * display_res_x) + xmin;
20513:S      6
20514:C      6       tymin := (dymin * display_res_y) + ymin;
20515:C      6       tymax := (dymax * display_res_y) + ymin;
20516:C      6     end;
20517:S
20518:C      5     { make sure new logical limits are within the physical limits }
20519:C      5     if (txmin >= xmin) and (txmax <= xmax + eight_diget_epsilon) and
20520:C      6     (tymin >= ymin) and (tymax <= ymax + eight_diget_epsilon) then
20521:C      6     begin
20522:C      6       display_limits (txmin,txmax,tymin,tymax);
20523:S      6
20524:C      6       { set flag indicating that the char size needs to be
20525:C      6         recalculated. This is done since the physical character size
20526:C      6         may change due to this procedure }
20527:S      6
20528:C      6       calc_text_xform := true;
20529:S
20530:C      6       ierr := 0;
20531:C      6     end
20532:C      6     else ierr := 2;
20533:C      5   end
20534:C      5   else
20535:C      3     ierr := 1;
20536:C      2 end; ( set_display_lim )
20537:S
20538:D      1 procedure clear_display;
20539:D      1
20540:S      2 ( Purpose : To clear to display      )
20541:D      2
20542:S      2 begin
20543:C      2   ck_system_init;
20544:C      2   ck_display_init;
20545:C      2
20546:S      2   with gle_gcb^ do
20547:C      3   begin
20548:C      3     info1 := -1; { clear all planes }
20549:C      3     info2 := gcb^.dgl_background_index;
20550:C      3     gle_clear ( gle_gcb );
20551:C      3   end;
20552:C      3

```



```

20553:C      2 end; ( Clear_display )
20554:S
20555:S
20556:D      1 $stackcheck off$
20557:D      -16 1 procedure move (x, y : real);
20558:S
20559:D      -16 2 ( Purpose : To change the current position
20560:S
20561:C      2 begin
20562:C      2   if disp_init then
20563:C      3   begin
20564:C      3     { save world cp }
20565:C      3     int_cp := false; { cp saved as real value }
20566:C      3     world_real_cpx := x;
20567:C      3     world_real_cpy := y;
20568:S
20569:C      3     { calc new device dependent cp }
20570:C      3     cpx := trunc ( x * xwtod_scale + xwtod_offset );
20571:C      3     cpy := trunc ( y * ywtod_scale + ywtod_offset );
20572:S
20573:C      3     {WRITELN('MOVE (' ,CPX:S,',',CPY:S,')');}
20574:C      3     with gle_gcb^ do
20575:C      4     begin
20576:C      4       end_x := cpx;
20577:C      4       end_y := cpy;
20578:C      4       call (move,gle_gcb);
20579:C      4     end;
20580:C      3   end
20581:C      3   else
20582:C      3     if system_init then error (err_dis_int)
20583:C      4     else error (err_sys_int);
20584:C      2 end; ( move )
20585:S
20586:S
20587:D      -16 1 procedure line (x, y : real);
20588:S
20589:D      -16 2 ( Purpose : To draw a line
20590:S
20591:C      2 begin
20592:C      2   if disp_init then
20593:C      3   begin
20594:C      3     { save world cp }
20595:C      3     int_cp := false; { cp saved as real value }
20596:C      3     world_real_cpx := x;
20597:C      3     world_real_cpy := y;
20598:S
20599:C      3     { calc new device dependent cp }
20600:C      3     cpx := trunc ( x * xwtod_scale + xwtod_offset );
20601:C      3     cpy := trunc ( y * ywtod_scale + ywtod_offset );
20602:S
20603:C      3     {WRITELN('LINE (' ,CPX:S,',',CPY:S,')');}
20604:C      3     with gle_gcb^ do
20605:C      4     begin
20606:C      4       end_x := cpx;
20607:C      4       end_y := cpy;
20608:C      4       call (draw,gle_gcb);
20609:C      4     end;
20610:C      3   end
20611:C      3   else
20612:C      3     if system_init then error (err_dis_int)

```

```

20613:C      4   else error (err_sys_int);
20614:C      2 end; ( line )
20615:S
20616:D      1 procedure int_move (ix, iy : gshortint);
20617:S
20618:D      2 ( Purpose : To move the current position
20619:S
20620:C      2 begin
20621:C      2   if disp_init then
20622:C      3   begin
20623:C      3     { use the normal move routine unless the short flag is set }
20624:C      3     if short_flag then
20625:C      4     begin
20626:C      4       { save world cp }
20627:C      4       int_cp := true; { cp saved as gshortint }
20628:C      4       world_int_cpx := ix;
20629:C      4       world_int_cpy := iy;
20630:S
20631:C      4       with gle_gcb^ do
20632:C      5       begin
20633:C      5         end_x := ix;
20634:C      5         end_y := iy;
20635:C      5         dgl_scaled_move; { perform a scaled move }
20636:C      5       end;
20637:C      4     end
20638:C      4     else move(ix,iy);
20639:C      3   end
20640:C      3   else
20641:C      3     if system_init then error (err_dis_int)
20642:C      4     else error (err_sys_int);
20643:C      2 end; ( int_move )
20644:S
20645:S
20646:D      1 procedure int_line (ix, iy : gshortint);
20647:S
20648:D      2 ( Purpose : To set the logical display limits
20649:S
20650:S
20651:C      2 begin
20652:C      2   if disp_init then
20653:C      3   begin
20654:C      3     { use normal line unless short_flag is set }
20655:C      3     if short_flag then
20656:C      4     begin
20657:C      4       { save world cp }
20658:C      4       int_cp := true; { cp saved as gshortint }
20659:C      4       world_int_cpx := ix;
20660:C      4       world_int_cpy := iy;
20661:C      4       with gle_gcb^ do
20662:C      5       begin
20663:C      5         end_x := ix;
20664:C      5         end_y := iy;
20665:C      5         dgl_scaled_draw; { perform a scaled draw }
20666:C      5       end;
20667:C      4     end
20668:C      4     else
20669:C      4       line(ix,iy);
20670:C      3   end
20671:C      3   else
20672:C      3     if system_init then error (err_dis_int)

```



```

20673:C      4      else error (err_sys_int);
20674:C      2 end; { int_line }
20675:S
20676:D      1 procedure int_polyline ( num_points : integer;
20677:D      2      anyvar xvec, yvec : gshortint_list );
20678:D      2 var
20679:D      -4      1 : integer;
20680:S
20681:S      2 begin
20682:C      2      ck_system_init;
20683:C      2      ck_display_init;
20684:S
20685:C      2      if num_points <= 0 then error(err_neg_points);
20686:S
20687:C      2      int_move ( xvec[1], yvec[1] );
20688:C      2      for i := 2 to num_points do
20689:C      3          int_line ( xvec[i], yvec[i]);
20690:C      2 end;
20691:S
20692:D      1 procedure polyline ( num_points : integer;
20693:D      2      anyvar xvec, yvec : greal_list );
20694:D      2 var
20695:D      -4      1 : integer;
20696:S
20697:S      2 begin
20698:C      2      ck_system_init;
20699:C      2      ck_display_init;
20700:S
20701:C      2      if num_points <= 0 then error(err_neg_points);
20702:S
20703:C      2      move ( xvec[1], yvec[1] );
20704:C      2      for i := 2 to num_points do
20705:C      3          line ( xvec[i], yvec[i]);
20706:C      2 end;
20707:D      1 $stackcheck on$
20708:S
20709:D      -256 1 procedure gtext( s : gstring255 );
20710:S
20711:D      -256 2 { PURPOSE : To draw a text string
20712:S
20713:S      2 begin
20714:C      2      if disp_init then
20715:C      3          begin
20716:C      3              if calc_text_xform then
20717:C      4                  with gcb^ do
20718:C      5                      begin
20719:C      5                          set_char_size (dgl_char_width, dgl_char_height);
20720:C      5                          set_text_rot (char_rot_w, char_rot_h);
20721:C      5                          calc_text_xform := false;
20722:C      5                      end;
20723:C      3              with gle_gcb^ do
20724:C      4                  begin
20725:C      4                      info_ptr1 := addr(s[1]);
20726:C      4                      info1 := strlen(s);
20727:C      4                  end;
20728:C      3              gle_text ( gle_gcb );
20729:C      3          end
20730:C      3      else
20731:C      3          if system_init then error (err_dis_int)
20732:C      4          else error (err_sys_int);

```

```

20733:C      2 end; { gtext }
20734:S
20735:D      1 procedure marker ( marker_number : integer );
20736:S
20737:S      2 begin
20738:C      2      ck_system_init;
20739:C      2      ck_display_init;
20740:C      2      if (marker_number < 1) or (marker_number > 19) then marker_number := 1;
20741:S
20742:C      2      with gle_gcb^ do
20743:C      3          begin
20744:C      3              info1 := marker_number;
20745:C      3              gle_set_marker ( gle_gcb );
20746:C      3              gle_marker ( gle_gcb );
20747:C      3          end;
20748:C      2 end;
20749:S
20750:D      1 procedure input_esc (      opcode : integer;
20751:D      2      isize : integer;
20752:D      2      rsize : integer;
20753:D      2      anyvar ilist : gint_list;
20754:D      2      anyvar rlist : greal_list;
20755:D      2      var ierr : integer );
20756:S
20757:D      2 { Purpose : To perform an input escape function
20758:S
20759:S      2 begin
20760:C      2      ck_system_init;
20761:C      2      ck_display_init;
20762:S
20763:C      2      ierr := opcode_ck ( opcode, isize, rsize );
20764:C      2      call (gcb^.proc_input_esc, opcode, isize, rsize, ilist, rlist, ierr);
20765:S
20766:C      2 end; { input_esc }
20767:S
20768:D      1 procedure output_esc (      opcode : integer;
20769:D      2      isize : integer;
20770:D      2      rsize : integer;
20771:D      2      anyvar ilist : gint_list;
20772:D      2      anyvar rlist : greal_list;
20773:D      2      var ierr : integer );
20774:S
20775:D      2 { Purpose : To perform an output escape funtion
20776:S
20777:S      2 begin
20778:C      2      ck_system_init;
20779:C      2      ck_display_init;
20780:S
20781:C      2      ierr := opcode_ck ( opcode, isize, rsize );
20782:C      2      call (gcb^.proc_output_esc, opcode, isize, rsize, ilist, rlist, ierr);
20783:C      2 end; { output_esc }
20784:S
20785:D      -16 1 procedure set_text_rot (dx, dy : real);
20786:S
20787:D      -16 2 { PURPOSE : To set the new text rotation vectors
20788:S
20789:D      -16 2 { calc normalize vector and save
20790:D      -16 2 { set flag so text xform is recalculated
20791:S
20792:D      -16 2 var

```



```

20793:D -24 2 r : real;
20794:C
20795:C 2 begin
20796:C 2 ck_system_init;
20797:C 2 ck_display_init;
20798:S
20799:C 2 if (dx = 0) and (dy = 0) then error (err_bad_parms);
20800:S
20801:C 2 with gcb^ do
20802:C 2 begin
20803:C 3 r := sqrt ( dx*dx + dy*dy);
20804:C 3 char_rot_w := dx / r;
20805:C 3 char_rot_h := dy / r;
20806:C 3 with gle_gcb^ do
20807:C 3 begin
20808:C 4 info1 := trunc(char_rot_w * 32768);
20809:C 4 info2 := trunc(char_rot_h * 32768);
20810:C 4 end;
20811:C 3 gle_text_dir ( gle_gcb );
20812:C 3 end
20813:C 3 end; { set_text_rot }
20814:S
20815:S
20816:D -16 1 procedure set_char_size (width,height:real);
20817:C
20818:C -16 2 { PURPOSE : To set the new character size }
20819:S
20820:D -16 2 { save width and height, set flag so text xform is recalculated }
20821:S
20822:C 2 begin
20823:C 2 ck_system_init;
20824:C 2 ck_display_init;
20825:S
20826:C 2 {if (width = 0.0) or (height = 0.0) then error (err_bad_parms);}
20827:S
20828:C 2 with gcb^ do
20829:C 2 begin
20830:C 3 dgl_char_width := width;
20831:C 3 dgl_char_height := height;
20832:C 3 with gle_gcb^ do
20833:C 3 begin
20834:C 4 info1 := abs(trunc((width * xwtod_scale * 7 / 9) * 8));
20835:C 4 info2 := abs(trunc((height * ywtod_scale * 9 / 15) * 8));
20836:C 4 gle_char_size ( gle_gcb );
20837:S
20838:C 4 info1 := abs(trunc((width * xwtod_scale * 2 / 9) * 8));
20839:C 4 info2 := abs(trunc((height * ywtod_scale * 6 / 15) * 8));
20840:C 4 gle_text_spacing ( gle_gcb );
20841:C 4 end;
20842:C 3 end
20843:C 3 end; { set_char_size }
20844:S
20845:S
20846:D -16 1 procedure set_echo_pos (wx,wy : real);
20847:C
20848:C -16 2 { Purpose : To set the locator echo position }
20849:S
20850:D -24 2 function between ( x1, p, x2 : real ) : boolean; { added for 2.1 bug fix }
20851:S
20852:C 3 begin

```

```

20853:C 3 between := ((x1 <= p) and ( p <= x2)) or
20854:C 3 ((x2 <= p) and ( p <= x1));
20855:C 3 end;
20856:S
20857:C 2 begin
20858:C 2 ck_system_init;
20859:C 2 ck_display_init;
20860:C 2 ck_locator_init;
20861:S
20862:C 2 with gcb^ do
20863:C 2 with window_lim do
20864:C 2 begin
20865:C 3 { ck bounds }
20866:C 3 if (between (xmin,wx,xmax) and { 2.1 bug fix }
20867:C 3 between (ymin,wy,ymax)) then
20868:C 3 begin
20869:C 4 { set world coord echo pos }
20870:C 4 w_loc_echo_x := wx;
20871:C 4 w_loc_echo_y := wy;
20872:C 4 { convert to display units }
20873:C 4 convert_wtod (w_loc_echo_x,w_loc_echo_y,d_loc_echo_x,d_loc_echo_y);
20874:C 4 end
20875:C 3 { ignor call if outside window }
20876:C 3 end
20877:C 2 end; { set_echo_pos }
20878:C
20879:S
20880:D -32 2 procedure set_locator_lim ( lxmin,lxmax,lymin,lymax : real;
20881:D var ierr : integer );
20882:S
20883:S
20884:D -32 2 { Purpose : To set the locator echo position }
20885:S
20886:D -32 2 var
20887:D 2 lxmin : real;
20888:D 2 lxmax : real;
20889:D 2 lymin : real;
20890:D 2 lymax : real;
20891:S
20892:C 2 begin
20893:C 2 ck_system_init;
20894:C 2 ck_locator_init;
20895:S
20896:C 2 { input limits can only be changed if the input device is not the same }
20897:C 2 physical device as the display }
20898:C 2 with gcb^ do
20899:C 2 if not disp_eq_loc then
20900:C 2 begin
20901:C 3 { ck parms }
20902:C 3 if (lymin < lxmax) and (lymin < lymin) then
20903:C 3 with gle_gcb^ do
20904:C 3 with max_loc_lim do
20905:C 3 begin
20906:C 4 { convert limits form mil to locator cord }
20907:C 4 lxmin := ((lxmin * input_res_x) + xmin);
20908:C 4 lxmax := ((lxmax * input_res_x) + xmin);
20909:C 4 lymin := ((lymin * input_res_y) + ymin);
20910:C 4 lymax := ((lymax * input_res_y) + ymin);
20911:S
20912:C 4 { make sure new logical limits are within the physical

```



```

20913:C      7      limits
20914:C      7      if (txmin >= xmin) and
20915:C      8      {txmax <= xmax + eight_diget_epsilon) and
20916:C      8      {tymin >= ymin) and
20917:C      8      {tymax <= ymax + eight_diget_epsilon) then
20918:C      8      begin
20919:C      8      { set new limits }
20920:C      8      locator_limits (txmin,txmax,tymin,tymax);
20921:C      8      ierr := 0;
20922:C      8      end
20923:C      8      else { bad limits }
20924:C      8      ierr := 2;
20925:C      7      end
20926:C      7      else { bad parms }
20927:C      5      ierr := 1;
20928:C      4      end
20929:C      4      else { locator and display are same device }
20930:C      4      ierr := 3;
20931:C      2 end; { set_locator_lim }
20932:C
20933:S
20934:D      1 procedure sample_locator( echo : integer;
20935:S      2      var rx,ry : real );
20936:S
20937:D      2 { Purpose : To sample the locator device
20938:S      }
20939:C      2 begin
20940:C      3   if loc_init then
20941:C      3   begin
20942:C      3   if disp_init then make_pic_current;
20943:C      3   call (gcb^.proc_sample_locator,echo,rx,ry);
20944:C      3   end
20945:C      3   else
20946:C      3   if system_init then error(err_loc_int)
20947:C      4   else error(err_sys_int);
20948:S
20949:C      2 end; { sample_locator }
20950:S
20951:D      1 procedure await_locator( echo : integer;
20952:D      2      var button : integer;
20953:D      2      var rx,ry : real );
20954:S
20955:S
20956:D      2 { Purpose : To activate the locator, and wait for operator termination
20957:S      }
20958:S
20959:D      2 var
20960:D      2   saved_pattern,
20961:D      2   saved_linewidth,
20962:D      2   saved_linestyle,
20963:D      2   saved_length,
20964:D      2   saved_mode,
20965:D      -24 2   saved_drawing_mode : integer;
20966:S
20967:C      2 begin
20968:C      2   ck_system_init;
20969:C      2   ck_locator_init;
20970:C      2   if disp_init then
20971:C      4   with gcb^,gle_gcb^ do
20972:C      4   begin

```

```

20973:C      4   make_pic_current;
20974:C      4   saved_pattern := current_linestyle_pattern;
20975:C      4   saved_linewidth := current_linewidth;
20976:C      4   saved_linestyle := current_linestyle;
20977:C      4   saved_length := current_pattern_length;
20978:C      4   saved_mode := current_linestyle_mode;
20979:C      4   saved_drawing_mode := current_drawing_mode;
20980:C      4
20981:C      4   info1 := 0;
20982:C      4   info2 := 0;
20983:C      4   info3 := 0;
20984:C      4   info4 := -1;
20985:C      4   gle_linestyle ( gle_gcb );
20986:C      4
20987:C      4   info1 := 1;
20988:C      4   gle_linewidth ( gle_gcb );
20989:S      4
20990:C      4   call (proc_color,cursor_color);
20991:C      4
20992:C      4   info1 := complement_mode;
20993:C      4   gle_define_drawing_mode ( gle_gcb );
20994:C      4   end;
20995:S
20996:C      2 call (gcb^.proc_await_locator,echo,button,rx,ry);
20997:S
20998:C      2 if disp_init then
20999:C      3   begin
21000:C      4   with gcb^,gle_gcb^ do
21001:C      4   begin
21002:C      4   info1 := saved_linestyle;
21003:C      4   info2 := saved_length;
21004:C      4   info3 := saved_mode;
21005:C      4   info4 := saved_pattern;
21006:C      4   gle_linestyle ( gle_gcb );
21007:S      4
21008:C      4   info1 := saved_linewidth;
21009:C      4   gle_linewidth ( gle_gcb );
21010:S      4
21011:C      4   info1 := saved_drawing_mode;
21012:C      4   gle_define_drawing_mode ( gle_gcb );
21013:S      4
21014:C      4   call (proc_color,dgl_current_color);
21015:C      4   end;
21016:S
21017:C      3   if echo > 1 then
21018:C      4   if not int_cp then move { world_real_cpx, world_real_cpy }
21019:C      5   else
21020:C      3   int_move { world_int_cpx, world_int_cpy };
21021:S
21022:C      2 end; { await_locator }
21023:S
21024:D      1 procedure display_term;
21025:S
21026:D      2 { Purpose: To terminate the graphics display
21027:S      }
21028:C      2 begin
21029:C      2   ck_system_init;
21030:C      2   ck_display_init;
21031:S
21032:C      2   with gcb^ do

```



```

21033:C      3      begin
21034:C      3          disp_init := false;
21035:C      3          disp_file_name := '';
21036:C      3          disp_dev_addr := 0;
21037:S
21038:C      3      { reset display limits to default }
21039:S
21040:C      3      with init_display_lim do
21041:C      4          display_limits(xmin,xmax,ymin,ymax);
21042:S
21043:C      3      if disp_eq_loc then with def_loc_lim do { locator limits no longer }
21044:C      5          locator_limits (xmin,xmax,ymin,ymax); { are effected by the display }
21045:S
21046:C      3      disp_eq_loc := not loc_init; { if both disabled then they are equal }
21047:S
21048:C      3      try
21049:C      4          call {gcb^.proc_color,0};
21050:C      4          gle_flush_buffer ( gle_gcb );
21051:C      4          gle_get_plp2 ( gle_gcb ); { Force read from device. This syncs
21052:C      4          gle_term(gle_gcb); { the OS with buffered devices }
21053:C      4          recover
21054:C      4          { ignore timeout errors }
21055:C      4          if (escapecode <= -26) or (loc_result <= 17) then escape(escapecode);
21056:C      4
21057:C      3      { dispose of color table and polygon table }
21058:C      3      if color_table_size > 0 then hpm_dispose(color_table_ptr,
21059:C      4          (color_table_size+1) * 24);
21060:C      3      if number_polygon_styles > 0 then nrm_dispose(poly_table_ptr,
21061:C      4          number_polygon_styles * 18 );
21062:C      3      end;
21063:C      3
21064:S
21065:C      2 end; { display_term }
21066:S
21067:D      1 procedure locator_term;
21068:S
21069:D      2 { Purpose: To terminate the locator device }
21070:S
21071:C      2 begin
21072:C      2      ck_system_init;
21073:C      2      ck_locator_init;
21074:S
21075:C      2      loc_init := false;
21076:C      2      with gcb^ do
21077:C      3          begin
21078:C      3              disp_eq_loc := not disp_init; { if both disabled then they are equal }
21079:C      3              loc_dev_addr := 0;
21080:C      3          end;
21081:S
21082:C      2      { reset to default locator limits }
21083:C      2      with init_locator_lim do
21084:C      3          locator_limits (xmin,xmax,ymin,ymax);
21085:S
21086:C      2      gle_input_term(gle_gcb);
21087:S
21088:C      2 end; { locator_term }
21089:S
21090:D      1 procedure graphics_term;
21091:S
21092:D      2 { Purpose: To terminate the graphics system }

```

```

21093:S
21094:C      2 begin
21095:C      2      ck_system_init;
21096:S
21097:C      2      { make sure all devices are terminated }
21098:C      2      if disp_init then display_term;
21099:C      2      if loc_init then locator_term;
21100:S
21101:C      2      { return dynamic vars to system }
21102:C      2      dispose(gcb);
21103:C      2      dispose(gle_gcb);
21104:C      2      dispose(gle_gcb);
21105:C      2      dispose(gle_knob_echo_gcb);
21106:C      2
21107:C      2      { set system initialized flag to disabled }
21108:C      2      system_init := false;
21109:C      2 end; { graphics_term }
21110:S
21111:D      1 procedure setup_display ( var ierr : integer);
21112:S
21113:D      2 var
21114:D      -4 2      i : integer;
21115:S
21116:D      -4 2      { Purpose: To set up display state after it has been initialized }
21117:S
21118:C      2 begin
21119:C      2      with gcb^ do
21120:C      3          begin
21121:C      3              with gle_gcb^ do
21122:C      4                  begin
21123:C      4                      max_disp_lim.xmin := display_min_x;
21124:C      4                      max_disp_lim.xmax := display_max_x;
21125:C      4                      max_disp_lim.ymin := display_min_y;
21126:C      4                      max_disp_lim.ymax := display_max_y;
21127:C      4
21128:C      4                      gle_get_plp2 ( gle_gcb );
21129:C      4
21130:C      4                      def_disp_lim.xmin := info1;
21131:C      4                      def_disp_lim.xmax := info2;
21132:C      4                      def_disp_lim.ymin := info3;
21133:C      4                      def_disp_lim.ymax := info4;
21134:C      4                  end;
21135:S
21136:C      3      disp_init := true;
21137:C      3      disp_eq_loc := ((disp_dev_addr = loc_dev_addr) or
21138:C      3      ((disp_dev_addr = internal_display) and
21139:C      3      (loc_dev_addr = internal_locator)));
21140:S
21141:C      3      { set up display limits }
21142:S
21143:C      3      with def_disp_lim do
21144:C      4          display_limits(xmin,xmax,ymin,ymax);
21145:S
21146:C      3      { set up default text size and rotation attributes }
21147:S
21148:C      3      dgl_char_width := init_char_width_factor *
21149:C      3      abs (window_lim.xmax - window_lim.xmin);
21150:C      3      dgl_char_height := init_char_height_factor *
21151:C      3      abs (window_lim.ymax - window_lim.ymin);
21152:C      3      set_char_size( dgl_char_width, dgl_char_height );

```



```

21153: 3
21154: 3 char_rot_w := init_char_rot_w;
21155: 3 char_rot_h := init_char_rot_h;
21156: 3
21157: 3 set_text_rot ( char_rot_w, char_rot_h );
21158: 3
21159: 3 ( set up all attributes here )
21160: 3
21161: 3 dgl_current_polygon_edge := true;
21162: 3 dgl_current_polygon_crosshatch := false;
21163: 3 dgl_current_polygon_linestyle := init_linestyle;
21164: 3 dgl_current_polygon_style := 1;
21165: 3 dgl_current_polygon_color := init_color;
21166: 3 dgl_polygon_color_current := false; ( color not set in gle )
21167: 3 dgl_current_polygon_density := 0;
21168: 3 dgl_current_polygon_angle := 0;
21169: 3 set_timing ( dgl_current_timing_mode );
21170: 3 dgl_current_color := -1; ( force calc of color )
21171: 3 set_color (init_color);
21172: 3 set_line_style (init_linestyle);
21173: 3 set_line_width (init_linewidth);
21174: 3
21175: 3 { init_cpy is in device units }
21176: 3 cpx := init_cpx;
21177: 3 cpy := init_cpy;
21178: 3
21179: 3 with gle_gcb^ do
21180: 4 begin
21181: 4 marker_size_x := trunc(display_res_x * 2.5 + 0.5); ( markers are 2.5 mm in size )
21182: 4 marker_size_y := marker_size_x;
21183: 4 info1 := marker_size_x;
21184: 4 info2 := marker_size_y;
21185: 4 gle_marker_size ( gle_gcb );
21186: 4
21187: 4 info1 := 1;
21188: 4 gle_graphics_on_off ( gle_gcb ); ( make sure graphics is on )
21189: 4 end;
21190: 3 end;
21191: 2 end; ( setup_display )
21192: 3
21193: 1 procedure display_finit ( fname : gstring255;
21194: 2 device_name : gstring255;
21195: 2 control : integer;
21196: 2 var ierr : integer );
21197: 3
21198: 3 { Purpose: To initialize the display device }
21199: 3
21200: 2 var
21201: 2 cnt : integer;
21202: 2
21203: 2 begin
21204: 2 ck_system_init;
21205: 2
21206: 2 ( make sure no display is currently enabled )
21207: 2 if disp_init then display_term;
21208: 2
21209: 2 if strlen(strltrim(strltrim(device_name))) <> 0 then
21210: 3 with gle_gcb^ do
21211: 4 begin
21212: 4 device_info := addr(fname[1]);

```

```

21213: 4 device_info_char_count := strlen(fname);
21214: 4 spooling := -1;
21215: 4 display_name := ' ';
21216: 4 display_name_char_count := min(strlen(device_name),6);
21217: 4 for cnt := 1 to display_name_char_count do
21218: 5 display_name[cnt] := device_name[cnt];
21219: 4 info1 := control;
21220: 4 info2 := 0; ( config DGL stuff )
21221: 4 configure_gle (gle_gcb);
21222: 4 ierr := error_return;
21223: 4 end
21224: 4 else
21225: 3 ierr := 2;
21226: 3
21227: 2 if ierr = 0 then
21228: 3 try
21229: 4 with gcb^ do
21230: 5 begin
21231: 5 disp_dev_addr := -1; ( indicate file name )
21232: 5 disp_file_name := fname;
21233: 5 setup_display ( ierr );
21234: 5 end;
21235: 4 recover;
21236: 4 begin
21237: 4 if escapecode = -20 then escape(escapecode); ( ignor all errors except stop key )
21238: 4 ierr := 2;
21239: 4 end
21240: 4 else
21241: 3 ierr := 2;
21242: 3
21243: 2 end; ( display_finit )
21244: 3
21245: 1 procedure display_init ( dev_addr : integer;
21246: 2 control : integer;
21247: 2 var ierr : integer );
21248: 3
21249: 3 { Purpose: To initialize the display device }
21250: 3
21251: 2 var
21252: 2 s : string[10];
21253: 2 cnt : integer;
21254: 2
21255: 2 begin
21256: 2 ck_system_init;
21257: 2
21258: 2 ( make sure no display is currently enabled )
21259: 2 if disp_init then display_term;
21260: 2
21261: 2 with gle_gcb^ do
21262: 3 begin
21263: 3 s := ' ';
21264: 3 strwrite(s,1,cnt,dev_addr,0);
21265: 3 device_info_char_count := strlen(s);
21266: 3 device_info := addr(s[1]);
21267: 3 spooling := 0;
21268: 3 info1 := control;
21269: 3 info2 := 0; ( config DGL stuff )
21270: 3 configure_gle (gle_gcb);
21271: 3 ierr := error_return;
21272: 3 end;

```



```

21273:S
21274:C 2 if ierr = 0 then
21275:C 3   try
21276:C 4     with gcb^ do
21277:C 5       begin
21278:C 6         disp_dev_adr := dev_adr;
21279:C 7         disp_file_name := '';
21280:C 8       end;
21281:C 9       setup_display ( ierr );
21282:C 4
21283:C 4       if gle_gcb^.complement_support = 1
21284:C 5         then gle_gcb^.cursor := dgl_cursor;
21285:C 4       recover
21286:C 4       begin
21287:C 4         if escapecode = -20 then escape(escapecode); ( ignor all errors exect stop key )
21288:C 4         ierr := 2;
21289:C 4       end
21290:C 4     else
21291:C 3       ierr := 2;
21292:S
21293:C 2 end; ( display_init )
21294:S
21295:D 1 procedure locator_init ( dev_adr : integer;
21296:D 2                             var ierr : integer );
21297:S
21298:D 2 ( Purpose: To initialize the locator device )
21299:S
21300:D 2 var
21301:D -12 s : string[10];
21302:D -16 i : integer;
21303:S
21304:C 2 begin
21305:C 2   ck_system_init;
21306:S
21307:C 2   ( make sure no locator is enabled )
21308:C 2   if loc_init then locator_term;
21309:S
21310:C 2   ( try to init a locator )
21311:S
21312:C 2   if disp_init then make_pic_current;
21313:S
21314:C 2   with gcb^,gle_gcbi^ do
21315:C 3     begin
21316:C 3       s := '';
21317:C 3       strwrite(s,1,1,dev_adr:0);
21318:C 3       device_info_char_count := strlen(s);
21319:C 3       device_info := addr(s[1]);
21320:C 3       info1 := 0;
21321:C 3       info2 := 0;
21322:C 3       configure_input_gle ( gle_gcbi );
21323:C 3       ierr := error_return;
21324:S
21325:C 3       if ierr = 0 then
21326:C 4         begin
21327:C 4           loc_init := true;
21328:C 4           loc_dev_adr := dev_adr;
21329:S
21330:C 4           with gle_gcbi^,gcb^.max_loc_lim do
21331:C 5             begin
21332:C 5               xmin := input_min_x;

```

```

21333:C 5       xmax := input_max_x;
21334:C 5       ymin := input_min_y;
21335:C 5       ymax := input_max_y;
21336:C 5     end;
21337:S
21338:C 4       gle_get_input_plp2 ( gle_gcbi );
21339:S
21340:C 4       with gle_gcbi^,gcb^,gcb^.def_loc_lim do
21341:C 5         begin
21342:C 5           xmin := info1;
21343:C 5           xmax := info2;
21344:C 5           ymin := info3;
21345:C 5           ymax := info4;
21346:C 5         end;
21347:S
21348:C 4       disp_eq_loc := disp_init and
21349:C 4         ((disp_dev_adr = loc_dev_adr) or
21350:C 4         ((disp_dev_adr = internal_display) and
21351:C 4         (loc_dev_adr = internal_locator)));
21352:S
21353:S ( If locator is not the same physical device as the graphics display,
21354:S then the locator limits are set to the default locator limits.
21355:S If the locator is the same device, then the locator limits
21356:C 4 are set to the current display limits. )
21357:S
21358:C 4       with gcb^ do
21359:C 5         begin
21360:C 5           if disp_eq_loc then
21361:C 6             with cur_disp_lim do locator_limits (xmin,xmax,ymin,ymax)
21362:C 7           else
21363:C 6             with def_loc_lim do locator_limits (xmin,xmax,ymin,ymax);
21364:C 5           end;
21365:C 4         end
21366:C 4       else
21367:C 4         ierr := 2;
21368:C 3       end;
21369:C 2 end; ( locator_init )
21370:S
21371:S 1 procedure graphics_init;
21372:S
21373:S 2 ( Purpose: To initialize the graphics system )
21374:D
21375:S
21376:C 2 begin
21377:C 2   ( make sure the system is not already init )
21378:C 2   if system_init then graphics_term;
21379:S
21380:C 2   ( set state flags )
21381:C 2   system_init := true;
21382:C 2   disp_init := false;
21383:C 2   loc_init := false;
21384:S
21385:S 2 ( get storage space -- changed from dynamic to global 2/84 BDS )
21386:C 2   gcb := addr(gcb_space);
21387:C 2   ( DGL high level storage )
21388:S
21389:C 2   gle_gcb := addr(gle_gcb_space);
21390:C 2   gle_init_gcb ( gle_gcb );
21391:C 2
21392:C 2   gle_gcbi := addr(gle_gcbi_space);
21393:C 2   ( locator input device )

```



```

21393:C      2   gle_init_input_gcb ( gle_gcbi );
21394:S
21395:C      2   gle_knob_echo_gcb := addr(gle_knob_echo_gcb_space);
21396:C      2   {knob echo output device (internal crt)}
21397:C      2   gle_init_gcb ( gle_knob_echo_gcb );
21398:S
21399:S      { kbdlangjumper is imported from sysgloables. Kata becomes true if
21400:C      2   the kata keyboard is installed }
21401:C      2   if kbdlang = katakana_kbd then gle_gcb^.kata := 1
21402:C      3   else gle_gcb^.kata := 0;
21403:S
21404:C      2   { set up defaults }
21405:C      2   with gcb^ do
21406:C      3   begin
21407:C      3   { When first inited that display and locator are the same device }
21408:C      3   disp_eq_loc := true;
21409:C      3   disp_dev_addr := init_dev_addr;
21410:C      3   disp_file_name := '';
21411:C      3   loc_dev_addr := init_dev_addr;
21412:C      3   window_lim := init_window;
21413:C      3   aspect_ratio := init_aspect;
21414:C      3   cur_vir_lim := init_vir_lim;
21415:C      3   viewport_lim := init_viewport;
21416:S
21417:C      3   { setup the default display/ locator limits to some large number }
21418:S
21419:C      3   with init_display_lim do
21420:C      4   display_limits (xmin, xmax, ymin, ymax );
21421:C      3   with init_locator_lim do
21422:C      4   locator_limits (xmin, xmax, ymin, ymax );
21423:S
21424:C      3   { explicitly set the cp to init_value }
21425:S
21426:C      3   cpx := init_cpx;
21427:C      3   cpy := init_cpy;
21428:C      3   int_cp := true;
21429:C      3   world_int_cpx := init_cpx;
21430:C      3   world_int_cpy := init_cpy;
21431:S
21432:C      3   { set up default text size and rotation attributes }
21433:S
21434:C      3   dgl_char_width := init_char_width;
21435:C      3   dgl_char_height := init_char_height;
21436:S
21437:C      3   char_rot_w := init_char_rot_w;
21438:C      3   char_rot_h := init_char_rot_h;
21439:S
21440:C      3   { set flag, indicating that the text xform needs to be recalculated }
21441:S
21442:C      3   calc_text_xform := true;
21443:S
21444:C      3   dgl_current_color := init_color;
21445:C      3   dgl_current_linestyle := init_linestyle;
21446:C      3   dgl_current_linewidth := init_linewidth;
21447:C      3   dgl_current_timing_mode := init_timing_mode;
21448:C      3   cursor_color := init_color;
21449:C      3   disp_just := centered;
21450:C      3   display_echo_mult := 1;
21451:C      3   graphics_error := 0;
21452:S

```

```

21453:C      3   number_polygon_styles := default_poly_table_size;
21454:C      3   color_table_size := default_color_table_size;
21455:C      3   dgl_current_polygon_color := 1;
21456:C      3   dgl_current_polygon_linestyle := 1;
21457:C      3   dgl_current_polygon_density := 0;
21458:C      3   dgl_current_polygon_angle := 90;
21459:C      3   dgl_current_polygon_edge := true;
21460:C      3   dgl_current_polygon_crosshatch := false;
21461:C      3   dgl_current_polygon_style := 1;
21462:C      3   dgl_current_color_model := 1;
21463:C      3   end;
21464:C      2 end; { graphics_init }
21465:S
21466:C      1 end. {module DGL_LIB}
21467:S
21468:D      1
21469:D      1 $LIST ON$
21470:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

LIBRARIAN

Description

The LIBRARIAN is a utility program used to create libraries, to link modules together or to disassemble code files.

Usage

The LIBRARIAN is invoked from the main command level by pressing "L".

Requirements

SYSGLOBALS, FS, ASM, MISC, LOADER, LDR, SYSDEVS, and CI.

Notes

The LIBRARIAN is sometimes called the "Linker"; however, it is not the same as the "linking loader" found in INITLOAD.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```
1:S      (*
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1984.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S     RESTRICTED RIGHTS LEGEND
11:S
12:S     Use, duplication, or disclosure by the Government
13:S     is subject to restrictions as set forth in
14:S     paragraph (b) (3) (E) of the Rights in Technical
15:S     Data and Computer Software clause in
16:S     DAR 7-104.9(a).
17:S
18:S     HEWLETT-PACKARD COMPANY
19:D     0 Fort Collins, Colorado      *)
20:S
21:S
22:D     0 $MODCAL,debug off,iocheck off,range off,ovflcheck off$ $ref 50$
23:S
24:D     0 program linker(input, output, keyboard);
25:S
26:S
27:D     1 import sysglobals,fs,loader,ldr,asm,sysdevs,ci,misc;
28:S
29:S
30:D     1 const    pagelines = 63;
31:D     1           pageblocks = 2;
32:D     1           entrysize = 26;
33:S
34:D     1 type     address = integer;
35:D     1           point = ^integer;
36:S
37:D     1 var      keyboard: text;
38:D     -2 1      todaysdate: daterec;
39:D     -4 1      linkerdate: daterec;
40:D     -18 1     tempstring: string[12];
41:D     -100 1    gvaluestring: string80;
42:D     -100 1    copyright:
43:D     -264 1    listfilename: string80;
44:D     -928 1    listing: text;
45:D     -928 1    pagenum:
46:D     -932 1    linenum: shortint;
47:D     -932 1    linking,booting,
48:D     -932 1    outopen,
49:D     -932 1    verifying,defsout,
50:D     -932 1    printopen,
51:D     -939 1    printeron: boolean;
52:D     -940 1    commandchar: char;
53:D     -944 1    startgvr: address;
54:D     -948 1    startgvrmod: moddescptr;
55:D     -952 1    modsave: moddescptr;
56:D     -956 1    infost: address;
57:S
58:D     -960 1    ires: integer;      (saved ioresult)
59:D     -964 1    errors: integer;
60:D     -968 1    esccode: integer;
```

```
61:D     -976 1    lowheap0,highheap0: address;
62:S
63:D     1058 1    infilename: string80;
64:D     1060 1    vmodnum: shortint;
65:S
66:D     1060 1    (output file information: )
67:D     1064 1    outdirectory: address;      (new library directory pointer)
68:D     1078 1    outfile: phyle;      (file being written)
69:D     1078 1    firstoutblock,
70:D     1078 1    outblock: integer;      (next block within library to write)
71:D     1078 1    nextblock: integer;      (next block within module to write)
72:D     1078 1    outfilename: string80;
73:D     1078 1    outmodnum: integer;      (number of modules created so far)
74:D     1078 1    outdirectsize,
75:D     1078 1    maxmodules: integer;
76:S
77:D     1834 1    (linker information: )
78:D     1838 1    totalpatchspace: integer;      (bytes of patch space)
79:D     1842 1    patchbytes: integer;
80:D     1842 1    backwardpatches,
81:D     1850 1    forwardpatches: moddescptr;
82:D     1854 1    newmodname: address;      (new name for linked module)
83:D     1858 1    infostart: address;      (pointer to bottom of linker memory)
84:D     1862 1    newexttable: address;      (location of new EXT table)
85:D     1866 1    newextsize: integer;      (size in bytes of EXT table)
86:D     1870 1    newdirectory: address;      (pointer to new module directory)
87:D     1874 1    loadgvr: gvrptr;
88:S
89:D     1 procedure dobeep;
90:D     2 begin beep;if streaming then escape(-1); end;
91:S
92:D     1 procedure errorline;
93:D     2 begin ires := ioresult; fgotoxy(output, 0, 22); write(bellchar, cteol); end;
94:S
95:D     1 procedure ioerror;
96:D     2 begin write(' ioresult = ',ires:1); escape(123); end;
97:S
98:D     -82 1 procedure getcommandchar(s:string80; var c:char);
99:D     2 begin
100:D     2   fgotoxy(output,0,23); write(s,cteol);
101:D     2   read(keyboard,c);
102:D     2   fgotoxy(output,0,22); writeln(cteol);write(cteol);
103:D     2   if (c>='a') and (c<='z') then c:=chr(ord(c)-32);
104:D     2   end;
105:S
106:D     1 procedure writedate(var f: text; date: daterec);
107:D     2 type months = packed array[0..35] of char;
108:D     2 const monthname = months['JanFebMarAprMayJunJulAugSepOctNovDec'];
109:D     -4 2 var i,j: shortint;
110:D     2 begin
111:D     2   with date do
112:D     3   begin
113:D     3     if (month in [1..12]) and (day>0)
114:D     4     and (year<100) then
115:D     4       begin { Valid date }
116:D     4         write(f, day:2, '-');
117:D     4         j := (month - 1) * 3;
118:D     4         for i := 1 to j+2 do write(f, monthname[i]);
119:D     4         write(f, '-',year:2);
120:D     4       end
```



```

121:C      4   else write(f, '<no date>'); { Invalid date }
122:C      3   end;
123:C      2 end; {datestring}
124:S
125:D      1 procedure gbytes(var p: integer; size: integer);
126:C      2 begin
127:C      2   p := lowheap.a;      lowheap.a := lowheap.a + size;
128:C      2   if lowheap.a > highheap.a then escape(112);
129:C      2 end;
130:S
131:D      1 procedure blockwrite(anyvar f: fib; anyvar obj: window; blocks,block: integer);
132:C      2 begin
133:C      2   call (f.am, addr(f), writebytes, obj, blocks*fbksize, block*fbksize);
134:C      2   if ioresult <> ord(inoerror) then escape(113);
135:C      2 end;
136:S
137:D      1 procedure readblocks(var f: fib; anyvar obj: window; size, block: integer);
138:C      2 begin
139:C      2   call (f.am, addr(f), readbytes, obj, size, block*fbksize);
140:C      2   if ioresult <> ord(inoerror) then escape(-10);
141:C      2 end;
142:S
143:D      1 procedure gvrstring(var gvp:gvrptr; var val:integer; pcrel,nores: boolean);
144:S
145:D      2 (*advances g past the GVR, adds any absolute part to VAL,
146:D      2   and constructs a string representing the GVR in gvaluestring *)
147:S
148:D      2 type
149:D      2   rpp = ^referenceptr;
150:D      2 var
151:D      2   Rcount:      shortint;
152:D      2   done:        boolean;
153:D      2   g:          gvrptr;
154:D      2   i:          integer;
155:S
156:D      2 procedure sign(sub: boolean);
157:C      3 begin
158:C      3   if sub then sappend(gvaluestring,'-')
159:C      3   else if strlen(gvaluestring)>0 then sappend(gvaluestring,'+');
160:C      3 end;
161:S
162:D      2 begin {gvrstring}
163:C      2   gvaluestring := '';      Rcount := 0;
164:C      2   repeat
165:C      3   if pcrel then g := loadgvr
166:C      4   else g := gvp;
167:C      3   if g <> NIL then with g^ do
168:C      4   begin
169:C      5   if longoffset then
170:C      6   g:=gvrptr(integer(g)+sizeof(generalvalue,true))
171:C      6   else
172:C      6   g:=gvrptr(integer(g)+sizeof(generalvalue,false));
173:C      6   if valueextend then
174:C      7   begin
175:C      8   if not pcrel then val:= val + veptr(g)^.value;
176:C      8   g:=gvrptr(integer(g)+sizeof(valueextension,sint));
177:C      8   end;
178:C      6   case primarytype of
179:C      6   absolute: {no more value};
180:C      6   relocatable: Rcount := Rcount + 1;

```

```

181:C      6   global: begin sign(false); sappend(gvaluestring,'Gbase'); end;
182:C      6   general:
183:C      6   begin
184:C      6   done := false;
185:C      6   repeat with rpp(g)^ do
186:C      7   begin
187:C      8   if adr=0 then
188:C      9   if op=addit then Rcount := Rcount + 1
189:C      10  else Rcount := Rcount - 1
190:C      10  else if adr=1 then
191:C      10  begin sign(op=subit); sappend(gvaluestring,'Gbase'); end
192:C      10  else
193:C      11  begin sign(op=subit);
194:C      11  if newmods^unresbits.bmp^[adr] or nores
195:C      11  then sappend(gvaluestring,symbolptr(newexttable+4*adr)^)
196:C      11  else sappend(gvaluestring,symbolptr(point(newexttable+4*adr)^)^);
197:C      10  end;
198:C      8   done := last;
199:C      8   g := gvrptr(integer(g)+sizeof(referenceptr));
200:C      8   end;
201:C      7   until done;
202:C      6   end; {general}
203:C      6   end; {primarytype cases}
204:C      6   if not pcrel then gvp := g;
205:C      6   end; {with g^}
206:C      6   pcrel := not pcrel;
207:C      6   until pcrel;
208:C      6   while Rcount <> 0 do
209:C      7   begin sign(Rcount<0); sappend(gvaluestring,'Rbase');
210:C      7   if Rcount < 0 then Rcount := Rcount + 1
211:C      7   else Rcount := Rcount - 1;
212:C      7   end;
213:C      6   if (val <> 0) or (strlen(gvaluestring)=0) then
214:C      7   begin
215:C      8   if val >= 0 then sign(false);
216:C      8   strwrite(gvaluestring,strlen(gvaluestring)+1,i,val:1);
217:C      8   end;
218:C      6   end; {gvrstring}
219:S
220:D      1 procedure printhead(var f: text);
221:D      1 var time: timerec;
222:D      1 begin
223:D      2 write(f,'Librarian [Rev. 3.0 ');
224:D      2 if ioresult <> 0 then
225:D      3 begin
226:D      4 printopen := false;
227:D      4 printeron := false;
228:D      4 escape(118);
229:D      4 end;
230:D      2 writedate(f, linkerdate);
231:D      2 write(f,',',':7);
232:D      2 writedate(f, todaydate);
233:D      2 system(time);
234:D      2 with time do write(f, hour:4,':',minute:2,':',centisecond div 100:2);
235:D      2 if pagenum > 0 then write(f,'page ':10,pagenum:1);
236:D      2 write(f);
237:D      2 writein(f);
238:D      2 end;
239:S
240:S

```



```

241:D 1 procedure pageject;
242:D 2 var i: integer;
243:C -4 2 begin
244:C 2 if linenum > 0 then page(listing);
245:C 2 linenum := 0;
246:C 2 end;
247:S
248:D 1 procedure list;
249:C 2 begin
250:C 2 if linenum >= pagelines then pageject;
251:C 2 if linenum = 0 then
252:C 3 begin
253:C 3 pagenum := pagenum + 1;
254:C 3 printhdr(listing);
255:C 3 linenum := linenum + 2;
256:C 3 end;
257:C 2 linenum := linenum + 1;
258:C 2 end;
259:S
260:D 1 procedure listln;
261:C 2 begin writeln(listing); linenum := linenum + 1;
262:C 2 end;
263:S
264:D 1 procedure quit;
265:D 2 var ch: char;
266:C -1 2 begin
267:C 2 if (outopen and (outmodnum>0)) or
268:C 2 if (booting and (outblock>0)) then
269:C 3 begin
270:C 3 errorline;
271:C 3 if booting then writeln('WARNING: You didn''t finish booting');
272:C 3 else writeln('WARNING: You didn''t ''Keep'' the output file.');
```

```

301:S
302:D 2 var
303:D -4 2 nilgvr: gvrptr;
304:D -5 2 dumped: boolean;
305:D -8 2 fortranflag: boolean;
306:D -8 2 rangetype: (norange, pcrange, linerange);
307:D -20 2 lowrange, highrange, lastline: integer;
308:D -24 2 decodestate, oldstate: decodestateype;
309:D -36 2 PC, tablePC, casecodestart: integer;
310:D -40 2 codecount: integer; {bytes left in current inbuf}
311:D -44 2 codeindex: integer; {next byte of code in inbuf}
312:D -48 2 instrsize: 0..10; {byte count of current instruction}
313:D -52 2 refgvr: address; reflin, refloc: address;
314:D -56 2 inbuf, refptr: address;
315:S
316:D 2 procedure dumpmod (mp:moddescptr);
317:D -4 3 var junkint: integer;
318:D -36 3 producername: string[30];
319:D -40 3 textstep: address;
320:S
321:C 3 begin (dumpmod);
322:C 3 dumped := true;
323:C 3 with mp^, directory.drp^ do
324:C 4 begin
325:C 4 pageject;
326:C 4 textstep := directory.a^ * sizeof(moduledirectory);
327:C 4 list; write(listing, 'MODULE ');
328:C 4 if strlen(textstep.syp^)=0 then write(listing, '(no name)');
329:C 4 else write(listing, textstep.syp^);
330:C 4 write(listing, ' Created ');
331:C 4 writedate(listing, date); writeln(listing);
332:C 4 list; write(listing, 'NOTICE: ');
333:C 4 if strlen(notice)=0 then writeln(listing, '(none)');
334:C 4 else writeln(listing, notice);
335:C 4 fortranflag := (producer = 'F');
336:C 4 case producer of
337:C 5 'M': producername := 'Modcal Compiler';
338:C 5 'P': producername := 'Pascal Compiler';
339:C 5 'L': producername := 'Librarian';
340:C 5 'F': producername := 'FORTRAN Compiler';
341:C 5 'B': producername := 'BASIC Compiler';
342:C 5 'A': producername := 'Assembler';
343:C 5 'C': producername := 'C Compiler';
344:C 5 'D': producername := 'Ada Compiler';
345:C 5 otherwise producername := ' ';
346:C 5 producername[2] := producer;
347:C 5 end;
348:C 4 list; write(listing, ' Produced by ', producername, ' of ');
349:C 4 writedate(listing, revision); writeln(listing);
350:C 4 if systemid = 0 then systemid := 1;
351:C 4 list; writeln(listing, ' Revision number ', systemid:1);
352:C 4 list; writeln(listing, ' Directory size ', directorysize:6, ' bytes');
353:C 4 list; writeln(listing, ' Module size ', modulesize:6, ' bytes');
354:C 4 junkint := strlen(textstep.syp^);
355:C 4 textstep.a := textstep.a + junkint * 2 - ord(odd(junkint));
356:C 4 if executable then
357:C 5 begin
358:C 5 startgvr := textstep;
359:C 5 junkint := 0;
360:C 5 gvrstring(textstep.gvp, junkint, false, false);
```



```

361:C      5      list; writeln(listing, ' Execution address ', gvaluestring);
362:C      5      end
363:C      5      else
364:C      5      begin
365:C      5      startgvr.gvp:=NIL;
366:C      5      list; writeln(listing, ' Module NOT executable');
367:C      5      end;
368:C      4      list; writeln(listing, ' Code base ', relocatablebase,
369:C      4      Size 'relocatablesize,' bytes');
370:C      4      list; writeln(listing, ' Global base ', globalbase,
371:C      4      Size 'globalsize,' bytes');
372:C      4      if extsize <= 8 then extsize := 0;
373:C      4      list; writeln(listing, ' EXT block ', extblock:3, Size 'extsize,
374:C      4      ' bytes');
375:C      4      list; writeln(listing, ' DEF block ', defblock:3, Size 'defsize,
376:C      4      ' bytes');
377:C      4      list;
378:C      4      if sourcesize <> 0 then
379:C      4      writeln(listing, ' EXPORT block ', sourceblock:3, Size '
380:C      5      sourcesize,' bytes')
381:C      5      else
382:C      5      writeln(listing, ' No EXPORT text');
383:C      4      list; writeln(listing, ' There are ', textrecords, ' TEXT records');
384:C      4      listln; listln;
385:C      4      end; (with mp^, directory^ )
386:C      3      end; (dumpmod)
387:S
388:D      2      procedure prepunassem;
389:D      3      var i: integer;
390:D      3      begin
391:D      3      modsave := newmods;
392:D      3      newmods := NIL;
393:D      3      infost := lowheap;
394:D      3      loadinfo(vmodnum, true, true);
395:D      3      with newmods^, directory.drp^ do
396:D      4      begin
397:D      4      newexttable := extaddr.a;
398:D      4      gbytes(unresbits.a, ((extsize div 4 + 15) div 16)*2);
399:D      4      for i := 2 to extsize div 4 - 1 do unresbits.bmp^[i] := false;
400:C      4      for i := 2 to listsize-1 do unresbits.bmp^[listaddr^[i] div 4] := true;
401:C      4      end;
402:C      3      if not dumped then dumpmod(newmods);
403:C      3      end;
404:S
405:D      2      procedure nextref;
406:C      3      begin
407:C      3      if refgvr.a < reflim then
408:C      4      if refgvr.gvp^.longoffset then
409:C      5      refloc:=refloc+refgvr.gvp^.long
410:C      5      else
411:C      5      refloc:=refloc+refgvr.gvp^.short;
412:C      3      end;
413:S
414:D      2      procedure listinstruction;
415:D      3      type
416:D      3      regtype = (D,R);
417:D      3      reange = 0..7;
418:D      3      siz = [bytesiz, wordsiz, longsiz, invalid];
419:S      3      opsize = array[bytesiz..longsiz] of string[3];
420:D      3      const

```

```

421:S
422:D      3      var
423:D      -10      hexout: packed array[0..4,0..3] of hex;
424:D      -11      firstline: boolean; { 1st line of current instruction? }
425:D      -14      bytesleft: 0..10; { to be listed in current instr }
426:D      -270      instrbuf: string[255]; { alpha form of instruction }
427:S
428:D      -270      instr: packed record case integer of
429:D      -270      1: (opcode: 0..15;
430:D      -270      case integer of
431:D      -270      1: (cond: 0..15;
432:D      -270      displ: -128..127);
433:D      -270      2: (regl: reange;
434:D      -270      opcode: 0..7;
435:D      -270      eamode: 0..7;
436:D      -270      eareg: reange);
437:D      -270      3: (dummy: 0..7;
438:D      -270      bit8: boolean;
439:D      -270      size: siz)
440:D      -270      );
441:D      -270      3: (w: shortint);
442:D      -270      4: (lb, rb: byte);
443:D      -272      end; (instr)
444:S
445:D      -272      ext: packed record
446:D      -272      case integer of
447:D      -272      1: (uwordext: 0..65535);
448:D      -272      2: (wordext: shortint);
449:D      -272      3: (longext: integer);
450:D      -272      4: (regclass: regtype;
451:D      -272      reg: reange;
452:D      -272      long: boolean;
453:D      -272      dummy: 0..7;
454:D      -272      byteext: -128..127);
455:D      -272      5: (mask: packed array [0..15] of boolean);
456:D      -276      end;
457:S
458:D      3      procedure emitint(val: integer);
459:D      -4      var i: integer;
460:D      4      begin
461:C      4      strwrite(instrbuf, strlen(instrbuf)+1, i, val:1);
462:C      4      end;
463:S
464:D      3      procedure comma;
465:C      4      begin sappend(instrbuf, ','); end;
466:S
467:D      3      procedure space;
468:C      4      begin sappend(instrbuf, ' '); end;
469:S
470:D      3      procedure printinstrword;
471:D      -4      var k: integer;
472:D      4      begin write(listing, ' ');
473:C      4      for k := 0 to 3 do
474:C      5      write(listing, hexout[(instrsize-bytesleft) div 2,k]);
475:C      4      bytesleft := bytesleft-2;
476:C      4      end;
477:S
478:D      3      procedure getinstrbytes(size: shortint);
479:C      4      begin
480:C      4      if codecount < size then escape(121);

```



```

481:C      4      moveleft(inbuf.stp^[codeindex],ext,size);
482:C      4      moveleft(ext,hexout[instrsize div 2],size);
483:C      4      instrsize := instrsize+size;
484:C      4      codeindex := codeindex+size;
485:C      4      codecount := codecount-size;
486:C      4      end;
487:C      3
488:D      3      procedure getinstruction;
489:C      4      begin
490:C      4      instrsize := 0;
491:C      4      getinstrbytes(2);      instr.w := ext.wordext;
492:C      4      end;
493:C      3
494:D      3      procedure defineword;
495:C      4      begin
496:C      4      instrbuf := 'dc.w ';
497:C      4      with instr do
498:C      5      begin
499:C      5      emitint(w);
500:C      5      while strlen(instrbuf) < 11 do space;
501:C      5      sappend(instrbuf,' or dc.b ');
502:C      5      emitint(lb); comma; emitint(rb);
503:C      5      while strlen(instrbuf) < 30 do space;
504:C      5      sappend(instrbuf,' or dc.b ','');
505:C      5      if (lb >= 32) and (lb < 127) then instrbuf[43] := chr(lb);
506:C      5      if (rb >= 32) and (rb < 127) then instrbuf[44] := chr(rb);
507:C      5      end;
508:C      4      end;
509:C      3
510:D      3      procedure extend(size: integer; pcrel: boolean);
511:D      4      var offset, location, PCtemp: integer;
512:C      4      begin
513:C      4      location := PC+instrsize;
514:C      4      if size = 1 then (byte extension)
515:C      5      begin
516:C      5      PCtemp := location + 1;
517:C      5      size := 2;
518:C      5      end
519:C      5      else PCtemp := location;
520:C      4
521:C      4      while (refgvr.a<reflim) and (refloc<PCtemp) do
522:C      5      begin
523:C      5      offset := 0;
524:C      5      gvstring(refgvr.gvp,offset,false,false);
525:C      5      nextref;
526:C      5      end;
527:C      4
528:C      4      getinstrbytes(size);
529:C      4      if odd(PCtemp) then offset := ext.byteext;
530:C      4      else if size = 2 then offset := ext.wordext;
531:C      4      else offset := ext.longext;
532:C      4      if pcrel then offset := offset + location;
533:C      4      if refloc=PCtemp then
534:C      5      begin
535:C      5      gvstring(refgvr.gvp,offset,pcrel,false);
536:C      5      nextref;
537:C      5      end
538:C      4      else
539:C      5      gvstring(nulgvr,offset,pcrel,false);
540:C      4      sappend(instrbuf,gvaluestring);

```

```

541:C      4      end;
542:C      3
543:D      3      procedure decode;
544:D      4      label 1,2;
545:D      4      type
546:D      4      opndtype = (source,dest);
547:D      4      regsymtype = array[regtype] of string[1];
548:D      4      extsizetype = array[bytesiz..longsiz] of 1..4;
549:D      4      arithoptype = array[0..13] of string[3];
550:C      4
551:D      4      condcodetype = array[0..15] of string[2];
552:C      4
553:D      4      const
554:D      4      SP = 7;
555:C      4
556:D      4      regsym = regsymtype['d','a'];
557:D      4      extsize = extsizetype[1,2,4];
558:C      4
559:D      4      concode = condcodetype['t','f','hi','ls','cc','cs','ne','eq',
560:D      4      'vc','vs','pl','mi','ge','li','gi','le'];
561:D      4      arithop = arithoptype['or','sub','cmp','and','add'];
562:C      4
563:C      3
564:D      3      procedure osize;
565:D      4      var size: siz;
566:C      4      begin size := instr.size;
567:C      4      if size = invalid then goto 2;
568:C      4      sappend(instrbuf,opsize[size]);
569:C      4      end;
570:C      3
571:D      3      procedure emitdir(regclass: regtype; reg: regrange);
572:C      4      begin if (regclass = A) and (reg = SP) then sappend(instrbuf,'sp')
573:C      5      else begin
574:C      5      sappend(instrbuf,regsym[regclass]);
575:C      5      setstrlen(instrbuf, strlen(instrbuf) + 1);
576:C      5      instrbuf[strlen(instrbuf)] := htoc[reg];
577:C      5      end;
578:C      4      end;
579:C      3
580:D      3      procedure emitardef(reg: regrange);
581:C      4      begin sappend(instrbuf,'(');
582:C      4      emitdir(A,reg);
583:C      4      sappend(instrbuf,')');
584:C      4      end;
585:C      3
586:D      3      procedure emitardisp(reg: regrange);
587:C      4      begin extend(2,false);
588:C      4      emitardef(reg);
589:C      4      end;
590:C      3
591:D      3      procedure emitpbstincr(reg: regrange);
592:C      4      begin emitardef(reg);
593:C      4      sappend(instrbuf,'+');
594:C      4      end;
595:C      3
596:D      3      procedure emitpredecr(reg: regrange);
597:C      4      begin sappend(instrbuf,'-');
598:C      4      emitardef(reg);
599:C      4      end;
600:C      3

```



```

601:D      4      procedure emitindx(pcrel: boolean);
602:C      5      begin
603:C      5      extend(1,pcrel); sappend(instrbuf, '(');
604:C      5      with instr, ext do
605:C      6      begin
606:C      6      if not pcrel then
607:C      7      begin emitdir(A, eareg); comma; end;
608:C      6      emitdir(regclass,reg);
609:C      6      if long then sappend(instrbuf, '.l');
610:C      7      else sappend(instrbuf, '.w');
611:C      6      end;
612:C      5      end;
613:S      4
614:D      4      procedure emitimm(val: integer);
615:C      5      begin
616:C      5      sappend(instrbuf, '#'); emitint(val);
617:C      5      end;
618:S      4
619:D      4      procedure immediate(fsize: siz);
620:C      5      begin
621:C      5      if fsize = invalid then goto 2,
622:C      5      sappend(instrbuf, '#');
623:C      5      extend(extsize(fsize),false);
624:C      5      end; {immediate}
625:S      4
626:D      4      procedure emitea(fsize: siz);
627:C      5      begin
628:C      5      with instr do
629:C      6      case eamode of
630:C      7      0: emitdir(D,eareg);
631:C      7      1: emitdir(A,eareg);
632:C      7      2: emitardir(eareg);
633:C      7      3: emitpostincr(eareg);
634:C      7      4: emitpredecr(eareg);
635:C      7      5: emitardisp(eareg);
636:C      7      6: emitindx(false);
637:C      7      7: case eareg of
638:C      8      0: extend(2,false);
639:C      8      1: extend(4,false);
640:C      8      2: extend(2,true);
641:C      8      3: emitindx(true);
642:C      8      4: immediate(fsize);
643:C      8      5..7: goto 2;
644:C      8      end; {case eareg}
645:C      7      end; {case eamode}
646:C      5      end; {emitea}
647:S      4
648:D      4      procedure opcode0;
649:C      5      ( bit, MOVEP, immediate, MOVES );
650:S      4
651:D      5      type bitoptype = array[siz] of string[5];
652:D      5      immoptype = array[0..6] of string[4];
653:D      5      const bitop = bitoptype['btst', 'bchg', 'bclr', 'bset', ''];
654:D      5      immop = immoptype['ori', 'andi', 'subi', 'addi', 'l', 'eori', 'cmpl'];
655:S      4
656:C      5      begin { opcode0 }
657:C      5      with instr do
658:C      6      if bit8 then
659:C      7      if eamode = 1 then
660:C      8      begin

```

```

661:C      8      if odd(opmode) then instrbuf := 'movep.l';
662:C      8      else instrbuf := 'movep.w';
663:C      8      if opmode <= 5 then
664:C      9      begin emitardisp(eareg);
665:C      9      comma; emitdir(D,reg1);
666:C      9      end
667:C      9      else begin
668:C      9      emitdir(D,reg1);
669:C      9      comma; emitardisp(eareg);
670:C      9      end;
671:C      8      end
672:C      8      else begin {dynamic bit}
673:C      8      instrbuf := bitop[siz];
674:C      8      emitdir(D,reg1);
675:C      8      comma; emitea(bytesiz);
676:C      8      end
677:C      8      else if reg1=4 then
678:C      8      begin instrbuf := bitop[siz];
679:C      8      immediate(bytesiz); comma;
680:C      8      emitea(bytesiz {invalid});
681:C      8      end
682:C      8      else { NOT bit8 }
683:C      8      begin instrbuf := immop[reg1];
684:C      8      if reg1=7 then
685:C      9      begin { moves }
686:C      9      instrbuf := 'moves'; osize;
687:C      9      getinstrbytes(2);
688:C      8      if ext.long then
689:C      10      begin emitdir(ext.regclass,ext.reg); comma; emitea(size);
690:C      10      end
691:C      10      else
692:C      10      begin emitea(size); comma; emitdir(ext.regclass,ext.reg);
693:C      10      end;
694:C      8      end
695:C      8      else
696:C      9      if (eamode=7) and (eareg=4) then
697:C      10      begin
698:C      10      space; immediate(wordsiz); comma;
699:C      10      if size = bytesiz then sappend(instrbuf, 'ccr');
700:C      10      else sappend(instrbuf, 'sr');
701:C      10      end
702:C      10      else begin
703:C      10      osize; immediate(size); comma; emitea(size);
704:C      10      end;
705:C      8      end;
706:C      5      end; {opcode0}
707:S      4
708:D      4      procedure move;
709:D      5      { opcodes 1..3: move byte, long, word }
710:D      5      var lsize: siz;
711:C      5      begin
712:C      5      with instr do
713:C      6      begin
714:C      6      case opcode of
715:C      7      1: lsize := bytesiz;
716:C      7      2: lsize := longsiz;
717:C      7      3: lsize := wordsiz;
718:C      7      end;
719:C      7      instrbuf := 'move';
720:C      6      if opmode=1 then sappend(instrbuf, 'a');

```



```

721:C      6      sappend(instrbuf,opsize[lsize]);
722:C      6      emittea(lsize); comma;
723:C      6      (kluge to make emittea emit destination)
724:C      6      eamode := opcode; eareg := regl;
725:C      6      if (eamode=7) and (eareg>1) then goto 2;
726:C      6      emittea(lsize (invalid));
727:C      6      end;
728:C      5      end; (move)
729:S
730:D      4      procedure opcode4;
731:D      5      type miscotype = array[0..7] of string[5];
732:D      5      type unoptype = array[0..5] of string[4];
733:D      5      const predecr = 4; ( eamode for predecrement )
734:D      5      miscop =
735:D      5      miscotype['reset','nop','stop','rte','rtd','rts','trapv','rtr'];
736:D      5      unop = unoptype['negx','clr','neg','not','','ist'];
737:D      -82 5      var regstring: string80;
738:S
739:D      5      procedure emitreglist
740:D      6      (optype: opndtype; predecr: boolean; var regstring: string80);
741:D      6      { emit register list to 'regstring' according to mask }
742:D      6      type
743:D      6      regmasksymtype = array[0..15] of string[2];
744:D      6      const
745:D      6      regmasksym = regmasksymtype
746:D      6      ['d0','d1','d2','d3','d4','d5','d6','d7',
747:D      6      'a0','a1','a2','a3','a4','a5','a6','a7'];
748:D      6      var
749:D      6      state: (start, (waiting for a '1')
750:D      6      open, (have seen a lone '1')
751:D      -2 6      cont);
752:D      -14 6      j,k,bitcount: integer;
753:S
754:D      6      procedure transition(b: boolean);
755:D      -2 7      var states: shortint;
756:C      7      begin
757:C      7      if b then
758:C      8      if optype = source then states := 6 else states := 5;
759:C      7      case state of
760:C      7      start: if b then
761:C      9      begin state := open;
762:C      9      sappend(regstring,regmasksym[bitcount]);
763:C      9      end;
764:C      8      open : if b then
765:C      9      begin state := cont;
766:C      9      sappend(regstring,'-');
767:C      9      end;
768:C      9      else begin state := start;
769:C      9      sappend(regstring,'/');
770:C      9      end;
771:C      8      cont : if not b then
772:C      9      begin state := start;
773:C      9      sappend(regstring,regmasksym[bitcount-1]);
774:C      9      sappend(regstring,'/');
775:C      9      end;
776:C      8      end; (case)
777:C      7      end; (transition)
778:S
779:C      6      begin (emitreglist)
780:C      6      getinstrbytes(2);

```

```

781:C      6      if ext.wordext = 0 then regstring := '(none) '
782:C      7      else
783:C      7      begin
784:C      7      state := start;
785:C      7      bitcount := 0; regstring := '';
786:C      7      if not predecr then
787:C      8      for j := 1 downto 0 do
788:C      9      begin
789:C      9      for k := 7 downto 0 do
790:C      10     begin transition(ext.mask[k+j*8]);
791:C      10     bitcount := bitcount+1;
792:C      10     end;
793:C      9     transition(false);
794:C      9     end;
795:C      9     else
796:C      8     for j := 0 to 1 do
797:C      9     begin
798:C      9     for k := 0 to 7 do
799:C      10     begin transition(ext.mask[k+j*8]);
800:C      10     bitcount := bitcount+1;
801:C      10     end;
802:C      9     transition(false);
803:C      9     end;
804:C      7     end;
805:C      6     if optype = source then
806:C      7     regstring[strlen(regstring)] := '-';
807:C      7     else setstrlen(regstring, strlen(regstring)-1);
808:C      6     end; (emitreglist)
809:S
810:D      5      procedure emitunop;
811:C      6      begin with instr do
812:C      7      begin
813:C      7      instrbuf := unop[regl]; osize;
814:C      7      emittea(size (invalid));
815:C      7      end;
816:C      6      end;
817:S
818:D      5      procedure emitsreg;
819:C      6      begin
820:C      6      with ext do
821:C      7      begin
822:C      7      if dummy<>0 then goto 2;
823:C      7      if not long then
824:C      8      begin
825:C      8      if byteext=0 then sappend(instrbuf,'sfc')
826:C      9      else if byteext=1 then sappend(instrbuf,'dfc')
827:C      10     else goto 2;
828:C      8     end;
829:C      8     else
830:C      8     begin
831:C      8     if byteext=0 then sappend(instrbuf,'usp')
832:C      9     else if byteext=1 then sappend(instrbuf,'vbr')
833:C      10     else goto 2;
834:C      8     end;
835:C      7     end;
836:C      6     end;
837:S
838:D      5      procedure jmpstates;
839:C      6      begin with instr do
840:C      7      case eamode of

```



```

841:C      8      2,5,6;;
842:C      8      7: if eareg>3 then goto 2;
843:C      8      otherwise goto 2;
844:C      8      end;
845:C      6      end;
846:C      6
847:C      6      begin {opcode4}
848:C      6      with instr do
849:C      6      if bit8 then
850:C      7      if ord(size) = 2 then
851:C      8      begin instrbuf := 'chk '
852:C      8      emitc(wordsiz); comma;
853:C      8      emitdir(D,reg1);
854:C      8      end
855:C      8      else begin instrbuf := 'lea '
856:C      8      emitc(invalid); comma;
857:C      8      emitdir(A,reg1);
858:C      8      end
859:C      8      else { NOT bit8 }
860:C      7      case reg1 of
861:C      8      0: if size=invalid then
862:C      9      begin instrbuf := 'move sr,'; emitc(wordsiz);
863:C      9      end
864:C      9      else emitunop;
865:C      9      1: if size=invalid then
866:C      9      begin instrbuf := 'move ccr,'; emitc(wordsiz);
867:C      9      end
868:C      9      else emitunop;
869:C      9      2: if size = invalid then
870:C      9      begin instrbuf := 'move '
871:C      9      emitc(wordsiz);
872:C      9      sappend(instrbuf, 'ccr');
873:C      9      end
874:C      9      else emitunop;
875:C      9      3: if size = invalid then
876:C      9      begin instrbuf := 'move '
877:C      9      emitc(wordsiz);
878:C      9      sappend(instrbuf, 'sr');
879:C      9      end
880:C      9      else emitunop;
881:C      9      4: case ord(size) of
882:C      9      0: begin instrbuf := 'rbsd '
883:C      9      emitc(bytesiz (invalid));
884:C      9      end;
885:C      9      1: if eamode = 0 then
886:C      10      begin instrbuf := 'swap '; emitdir(D,eareg);
887:C      10      end
888:C      10      else
889:C      10      begin instrbuf := 'pea '; emitc(invalid);
890:C      10      end;
891:C      9      2,3: if eamode = 0 then
892:C      10      begin instrbuf := 'ext '
893:C      10      sappend(instrbuf,opsize[pred(size)]);
894:C      10      emitdir(D,eareg);
895:C      10      end
896:C      10      else
897:C      10      begin
898:C      10      instrbuf := 'movem';
899:C      10      sappend(instrbuf,opsize[pred(size)]);
900:C      10      emitreglist(source,eamode=predec,regstring);

```

```

901:C      10      sappend(instrbuf,regstring);
902:C      10      emitc(invalid);
903:C      10      end;
904:C      9      end; {case size}
905:C      9      5: if size = invalid then
906:C      9      begin instrbuf := 'tas '
907:C      9      emitc(bytesiz (invalid));
908:C      9      end
909:C      9      else emitunop;
910:C      9      6: if size<longsiz then goto 2
911:C      9      else
912:C      9      begin instrbuf := 'movem';
913:C      9      sappend(instrbuf,opsize[pred(size)]);
914:C      9      emitreglist(dest,false,regstring);
915:C      9      emitc(invalid); comma;
916:C      9      sappend(instrbuf,regstring);
917:C      9      end;
918:C      8      7: if ord(size) = 2 then
919:C      9      begin instrbuf := 'jsr '
920:C      9      jmpstates;
921:C      9      emitc(invalid);
922:C      9      end
923:C      9      else if ord(size) = 3 then
924:C      10      begin instrbuf := 'jmp '
925:C      10      jmpstates;
926:C      10      emitc(invalid);
927:C      10      end
928:C      10      else
929:C      10      case eamode of
930:C      11      0,1:
931:C      11      begin instrbuf := 'trap '
932:C      11      emitimm(eareg+8*eamode);
933:C      11      if eareg + 8*eamode = 9 then
934:C      12      begin
935:C      12      comma; immediate(wordsiz);
936:C      12      comma; extend(4,true);
937:C      12      end
938:C      12      else
939:C      12      if eareg + 8*eamode = 1 then
940:C      13      begin comma; immediate(wordsiz);
941:C      13      end
942:C      13      else
943:C      13      if eareg + 8*eamode = 0 then
944:C      14      begin
945:C      14      comma; getinstrbytes(2);
946:C      14      lastline := ext.uwordext;
947:C      14      emitimm(lastline);
948:C      14      while strlen(instrbuf) < 20 do space;
949:C      14      sappend(instrbuf, 'COMPILED LINE NUMBER ');
950:C      14      emitint(lastline);
951:C      14      end;
952:C      11      end;
953:C      11      2: begin instrbuf := 'link '
954:C      11      emitdir(A,eareg); comma;
955:C      11      immediate(wordsiz);
956:C      11      end;
957:C      11      3: begin instrbuf := 'unlk '; emitdir(A,eareg);
958:C      11      end;
959:C      11      4,5:
960:C      11      begin instrbuf := 'move '

```



```
961:C      11      if eamode = 5 then sappend(instrbuf,'usp');
962:C      11      emitdir(A,eareg);
963:C      11      if eamode = 4 then sappend(instrbuf,'usp');
964:C      11      end;
965:C      11      6: begin
966:C      11      instrbuf := miscop[eareg];
967:C      11      if (eareg=2) or (eareg=4) then (stop)(rtd)
968:C      12      begin space; immediate(wordsiz);
969:C      12      end;
970:C      11      end;
971:C      11      7: begin { movec }
972:C      11      if ord(size)<>1 then goto 2;
973:C      11      instrbuf := 'movec';
974:C      11      getinstrbytes(2);
975:C      11      if eareg=2 then
976:C      12      begin
977:C      12      emitsreg; comma; emitdir(ext.regclass,ext.reg);
978:C      12      end
979:C      12      else
980:C      12      if eareg=3 then
981:C      13      begin
982:C      13      emitdir(ext.regclass,ext.reg); comma; emitsreg;
983:C      13      end
984:C      13      else goto 2;
985:C      11      end;
986:C      11      end; {case eamode}
987:C      8      end; {case reg1}
988:C      5      end; {opcode4}
989:S      4
990:D      4      procedure quick;
991:C      5      begin
992:C      5      with instr do if reg1 = 0 then emitimm(8)
993:C      7      else emitimm(reg1);
994:C      5      comma;
995:C      5      end;
996:S      5
997:D      4      procedure shift;
998:D      5      type
999:D      5      shiftoptype = array[0..7] of string[4];
1000:D      5      const
1001:D      5      shiftopt =
1002:D      5      shiftoptype['asr','lsr','roxr','ror','asl','lsl','roxl','rol'];
1003:C      5      begin
1004:C      5      with instr do
1005:C      6      if size = invalid then
1006:C      7      begin instrbuf := shiftopt[4*ord(bit8)+reg1];
1007:C      7      space; emittea(bytesiz {invalid});
1008:C      7      end
1009:C      7      else
1010:C      7      begin
1011:C      7      instrbuf := shiftopt[4*ord(bit8)+eamode mod 4];
1012:C      7      osize;
1013:C      7      if eamode div 4 = 1 then
1014:C      8      begin
1015:C      8      emitdir(D,reg1);
1016:C      8      comma;
1017:C      8      end
1018:C      8      else quick;
1019:C      7      emitdir(D,eareg);
1020:C      7      end;
```

```
1021:C      5      end; {shift}
1022:S      4
1023:C      4      begin {decode}
1024:C      4      with instr do
1025:C      5      case opcode of
1026:C      6      0: opcode0;
1027:C      6      1,2,3: move;
1028:C      6      4: opcode4;
1029:C      6      5: if size = invalid then
1030:C      7      begin
1031:C      7      if eamode = 1 then
1032:C      8      begin
1033:C      8      instrbuf := 'db';
1034:C      8      if cond = 1 then sappend(instrbuf,'ra')
1035:C      8      else sappend(instrbuf,condcode[cond]);
1036:C      8      space; emitdir(D, eareg);
1037:C      8      comma; extend(2,true);
1038:C      8      end
1039:C      8      else
1040:C      8      begin
1041:C      8      instrbuf := 's';
1042:C      8      sappend(instrbuf,condcode[cond]);
1043:C      8      space; emittea(bytesiz {invalid});
1044:C      8      end;
1045:C      7      end
1046:C      7      else
1047:C      7      begin
1048:C      7      if bit8 then instrbuf := 'subq'
1049:C      8      else instrbuf := 'addq';
1050:C      7      osize; quick;
1051:C      7      emittea(size {invalid});
1052:C      7      end;
1053:C      6      6: begin instrbuf := 'b';
1054:C      6      if cond = 0 then sappend(instrbuf,'ra')
1055:C      7      else if cond = 1 then sappend(instrbuf,'sr')
1056:C      8      else sappend(instrbuf,condcode[cond]);
1057:S      8
1058:C      8      if displ = 0 then
1059:C      7      begin space; extend(2,true); end
1060:C      7      else
1061:C      7      begin sappend(instrbuf,'s ');
1062:C      7      ext.longext := pc + 2 + displ;
1063:C      7      gvrstring(nilgvr,ext.longext,true,false);
1064:C      7      sappend(instrbuf,gvaluesstring);
1065:C      7      end;
1066:C      6      end;
1067:C      6      7: begin instrbuf := 'moveq';
1068:C      6      emitimm(displ); comma;
1069:C      6      emitdir(D,reg1);
1070:C      6      end;
1071:S      6
1072:S      6
1073:S      6
1074:C      6      8,9,11,12,13: begin
1075:C      6      instrbuf := arithop[opcode];
1076:C      6      if size=invalid then
1077:C      7      begin
1078:C      7      if odd(opcode) then
1079:C      8      begin
1080:C      8      sappend(instrbuf,'a');
```



```

1081:C      8      if bit8 then
1082:C      9      begin
1083:C      9      sappend(instrbuf, opsize[longsiz]);
1084:C      9      emitea(longsiz);
1085:C      9      end
1086:C      9      else begin
1087:C      9      sappend(instrbuf, opsize[wordsize]);
1088:C      9      emitea(wordsize);
1089:C      9      end;
1090:C      8      comma; emitdir(A, reg1);
1091:C      8      end
1092:C      8      else
1093:C      8      begin
1094:C      8      if opcode = 8 then instrbuf := 'div';
1095:C      8      else instrbuf := 'mul';
1096:C      8      if bit8 then sappend(instrbuf, 's ');
1097:C      9      else sappend(instrbuf, 'u ');
1098:C      8      emitea(wordsize); comma; emitdir(D, reg1);
1099:C      8      end
1100:C      8      end
1101:C      7      else if (not bit8) or (eamode > 1) or (opcode = 11) then
1102:C      8      begin
1103:C      8      if opcode = 11 then
1104:C      9      if bit8 then
1105:C      10     if eamode = 1 then
1106:C      11     begin
1107:C      11     sappend(instrbuf, 'm'); osize;
1108:C      11     emitpostincr(eareg); comma; emitpostincr(reg1);
1109:C      11     goto 1;
1110:C      11     end
1111:C      11     else instrbuf := 'eor';
1112:C      8     osize;
1113:C      8     if bit8 then begin emitdir(D, reg1); comma; emitea(osize);
1114:C      9     end
1115:C      9     else begin emitea(osize); comma; emitdir(D, reg1);
1116:C      9     end;
1117:C      8     end
1118:C      8     else
1119:C      8     begin
1120:C      8     if odd(opcode) then begin sappend(instrbuf, 'x'); osize; end
1121:C      9     else if opcode = 8 then instrbuf := 'sbcd';
1122:C      10    else if size = bytesiz then instrbuf := 'abcd';
1123:C      11    else begin
1124:C      11    instrbuf := 'exg';
1125:C      11    if eamode = 0 then
1126:C      12    begin emitdir(D, reg1); comma; emitdir(D, eareg) end
1127:C      12    else if opcode = 5 then
1128:C      13    begin emitdir(A, reg1); comma; emitdir(A, eareg) end
1129:C      13    else
1130:C      13    begin emitdir(D, reg1); comma; emitdir(A, eareg) end;
1131:C      11    goto 1;
1132:C      11    end;
1133:C      9    if eamode = 0 then
1134:C      9    begin emitdir(D, eareg); comma; emitdir(D, reg1);
1135:C      9    end
1136:C      9    else begin emitpredecr(eareg); comma; emitpredecr(reg1);
1137:C      9    end;
1138:C      8    end;
1139:C      6    end;
1140:C      6    14:shift;

```

```

1141:C      6      otherwise goto 2;
1142:C      6      end; (case)
1143:C      4      goto 1;
1144:C      4      2: begin defineword;
1145:C      4      if decodestate <> abscode then decodestate := consts;
1146:C      4      end;
1147:C      4      1: end; (decode)
1148:S
1149:D      3      procedure definecaseword;
1150:D      4      var savepc: integer;
1151:D      4      begin
1152:C      4      instrsize := 0;
1153:C      4      instrbuf := 'case jump';
1154:C      4      savepc := pc; pc := tablepc;
1155:C      4      extend(2, true);
1156:C      4      pc := savepc;
1157:C      4      end;
1158:S
1159:D      3      procedure decodestuff;
1160:D      4      label 1;
1161:D      4      var temp: integer;
1162:S
1163:D      4      procedure printprocboundary;
1164:D      5      label 1;
1165:D      5      var defaddr, deflimit, len, gvrbase: integer;
1166:D      5      var veloc: address;
1167:D      5      begin
1168:C      5      defaddr:=newmods^.defaddr.a;
1169:C      5      deflimit:=defaddr+newmods^.defsize;
1170:C      5      while defaddr < deflimit do
1171:C      6      begin
1172:C      6      len:=strlen(symbolptr(defaddr));
1173:C      6      len:=len*2+ord(odd(len));
1174:C      6      gvrbase:=defaddr+len;
1175:C      6      with gvrptr(gvrbase)^ do
1176:C      7      if primarytype = loadgvr^.primarytype then
1177:C      8      begin
1178:C      8      veloc.a:=gvrbase+sizeof(generalvalue, false);
1179:C      8      if veloc.vcp.value = PC then goto 1;
1180:C      8      end;
1181:C      6      defaddr:=defaddr+len+ord(symtableptr(defaddr)^[len+1]);
1182:C      6      end;
1183:C      5      listln;
1184:C      5      1: list; write(listing,
1185:C      5      '-----',
1186:C      5      '-----');
1187:C      5      if defaddr < deflimit then
1188:C      6      write(listing, symbolptr(defaddr));
1189:C      5      writeln(listing);
1190:C      5      end; (printprocboundary)
1191:S
1192:C      4      begin (decodestuff)
1193:C      4      1: case decodestate of
1194:C      5      consts:
1195:C      5      begin getinstruction;
1196:C      6      if (instr.w = 20054) (LINK A6)
1197:C      6      or (instr.w = 20033) (TRAP #1) then
1198:C      6      begin decodestate := code; decode;
1199:C      6      if (rangetype = norange) or
1200:C      7      ((rangetype = pcrange) and (PC >= lowrange) and (PC < highrange)) or

```



```
1201:C      7      ((rangetype = linerange) and (lastline >= lowrange) and (lastline < highrange))
1202:C      7      then printprocboundary;
1203:C      6      end
1204:C      6      else if (instr.w = 20217) {JMP long abs}
1205:C      7      or (instr.w = 24576) {BRA 16 bit} then
1206:C      7      decode
1207:C      7      else defineword;
1208:C      5      end;
1209:C      5      abscode;
1210:C      5      code:
1211:C      5      begin
1212:C      5      getinstruction;
1213:C      5      decode;
1214:C      6      if decodestate <> abscode then
1215:C      7      if instr.w = 20062 {UNLK A6} then decodestate := endofproc
1216:C      8      else if instr.w = 20219 {JMP pc indexed} then
1217:C      5      begin oldstate := code; decodestate := startcase end;
1218:C      5      end;
1219:C      5      startcase:
1220:C      5      begin
1221:C      5      tablePC := PC;
1222:C      5      definecaseword;
1223:C      5      casecodestart:=ext.wordext+PC;
1224:C      5      decodestate := casetable;
1225:C      5      end;
1226:C      5      casetable:
1227:C      5      begin
1228:C      6      if PC = casecodestart
1229:C      6      then begin decodestate := oldstate; goto 1 end
1230:C      6      else
1231:C      6      begin definecaseword;
1232:C      7      if not fortranflag then
1233:C      7      begin
1234:C      7      temp:=ext.wordext+tablePC;
1235:C      7      if temp<casecodestart then casecodestart := temp;
1236:C      6      end;
1237:C      5      end;
1238:C      5      endofproc:
1239:C      5      begin getinstruction; decode;
1240:C      6      if (instr.w = 20085 {RTS})
1241:C      6      or (instr.w div 8 = 2522 {JMP (An)}) then
1242:C      5      decodestate := const;
1243:C      5      end;
1244:C      5      end; {case}
1245:C      4      end; {decodestuff}
1246:C      3      begin {listinstruction}
1247:C      3      decodestuff;
1248:C      3      bytesleft := instrsize; firstline := true;
1249:C      3      if (rangetype = norange) or
1250:C      4      ((rangetype = pcrange) and (PC >= lowrange) and (PC < highrange)) or
1251:C      4      ((rangetype = linerange) and (lastline >= lowrange) and (lastline < highrange)) then
1252:C      4      repeat
1253:C      5      list;
1254:C      5      if firstline then write(listing,PC:8,' ')
1255:C      6      else write(listing,' '9 (17) );
1256:C      5      printinstrword;
1257:C      5      if bytesleft>0 then printinstrword
1258:C      5      else
1259:C      6      else
1260:C      6
```

```
1261:C      6      if firstline then write(listing,' '5);
1262:C      5      if firstline then
1263:C      6      begin writein(listing,' '9,instrbuf); firstline := false end
1264:C      5      else writein(listing);
1265:C      5      until bytesleft = 0;
1266:C      3      PC:=PC + instrsize;
1267:C      3      end; {listinstruction}
1268:C      2      procedure getcodeblocks;
1269:C      2      var junkint: integer;
1270:C      2      var pclimit: integer;
1271:C      2      var textstep: address;
1272:C      2      var textrecctr: shortint;
1273:C      2      begin
1274:C      3      with newmods^,directory.drp^ do
1275:C      3      begin
1276:C      4      textstep.a:=directory.a+sizeof(moduledirectory);
1277:C      4      junkint:=strlen(textstep.syp^);
1278:C      4      textstep.a := textstep.a+junkint+2-ord(odd(junkint));
1279:C      4      if executable then textstep.a := textstep.a + textstep.gvp^.short;
1280:C      4      textrecctr:=textrecords;
1281:C      4      while textrecctr > 0 do with textstep.tdp^ do
1282:C      6      begin
1283:C      6      textrecctr:=textrecctr-1;
1284:C      6      list; writeln(listing,'TEXT RECORD #');
1285:C      6      textrecctr:=textrecctr-1;
1286:C      6      list; writeln(listing,'TEXT start block ',textstart:4,
1287:C      6      Size ,textsize,' bytes');
1288:C      6      list; writeln(listing,' REF start block ',refstart:4,
1289:C      6      Size ,refsize,' bytes');
1290:C      6      textstep.a :=textstep.a+sizeof(textdescriptor);
1291:C      6      PC := 0;
1292:C      6      loadgvr := textstep.gvp;
1293:C      6      gvsstring(textstep.gvp,PC,false,false);
1294:C      6      gbytes(inbuf.a,textsize);
1295:C      6      readblocks(filefib.fbp^,inbuf.p^,textsize,fileblock+textstart);
1296:C      6      codeindex:=0; codecount:=textsize;
1297:C      6      gbytes(refptr.a,refsize);
1298:C      6      readblocks(filefib.fbp^,refptr.p^,refsize,fileblock+refstart);
1299:C      6      refgvr:=refptr;
1300:C      6      reflim:=refptr.a+refsize;
1301:C      6      refloc:=PC; nextref;
1302:C      6      pclimit := PC + textsize;
1303:C      6      list; writeln(listing,' LOAD address ',gvaluelstring);
1304:C      6      listin;
1305:C      6      while PC < pclimit do listinstruction;
1306:C      6      listin; listin;
1307:C      6      lowheap := inbuf;
1308:C      6      end;
1309:C      4      end; {with newmods^,directory^}
1310:C      3      end; {getcodeblocks}
1311:C      2      procedure listdefs;
1312:C      2      var
1313:C      3      len,val: integer;
1314:C      3      lim,pl: address;
1315:C      3      begin
1316:C      3      prepunassem;
1317:C      3
```



```

1321:C      3 with newmods^ do
1322:C      4 begin
1323:C      4 list; writeln(listing, ' DEF table of ',
1324:C      4 fdirectory^[vmodnum].dtid, ' ');
1325:C      4 listln;
1326:C      4 pl := defaddr;
1327:C      4 lim.a := pl.a + defsize;
1328:C      4 while pl.a < lim.a do
1329:C      5 begin
1330:C      5 len:=strlen(pl.syp^);
1331:C      5 list; write(listing, ' :S,pl.syp^',(30-len));
1332:C      5 pl.a := pl.a + len*2-ord(odd(len));
1333:C      5 val := 0;
1334:C      5 gvrstring(pl.gvp,val,false,false);
1335:C      5 writeln(listing,gvalustring);
1336:C      5 end;
1337:C      4 end;
1338:C      3 listln;
1339:C      3 newmods := modsave;
1340:C      3 lowheap := infost;
1341:C      3 end;
1342:S
1343:D      2 procedure listexts;
1344:D      3 var
1345:D      3 i: integer;
1346:D      3 pi: addrec;
1347:D      3 begin
1348:D      3 prepunassem;
1349:D      3 with newmods^ do
1350:D      4 begin
1351:D      4 list; writeln(listing, ' EXT table of ',
1352:D      4 fdirectory^[vmodnum].dtid, ' ');
1353:D      4 listln;
1354:D      4 for i:=2 to listsize-1 do if listaddr^[i] <> 0 then
1355:D      5 begin
1356:D      5 pl.a := extaddr.a + listaddr^[i];
1357:D      5 list; writeln(listing, ' :S,pl.syp^');
1358:D      5 end;
1359:D      4 end;
1360:D      3 listln;
1361:D      3 newmods := modsave;
1362:D      3 lowheap := infost;
1363:D      3 end;
1364:S
1365:D      2 procedure listtext;
1366:D      3 const pagesize = pageblocks * blocksize;
1367:D      3 var
1368:D      3 textbuf,ptr: addrec;
1369:D      3 i, pages: integer;
1370:D      3 readsize: integer;
1371:D      3 linestart: boolean;
1372:S
1373:D      3 procedure dochar(c: char);
1374:D      4 begin
1375:D      4 if linestart then list;
1376:D      4 listln;
1377:D      4 if linestart then writeln(listing) else write(listing, c);
1378:D      4 end;
1379:S
1380:C      3 begin

```

```

1381:C      3 prepunassem;
1382:C      3 gbytes(textbuf.a, pagesize);
1383:C      3 with newmods^, directory.drp^ do
1384:C      4 begin
1385:C      4 list; writeln(listing, ' DEFINE SOURCE of ',
1386:C      4 fdirectory^[vmodnum].dtid, ' ');
1387:C      4 listln;
1388:C      4 pages := (sourceize + (pagesize-1)) div pagesize;
1389:C      4 for j := 0 to pages-1 do
1390:C      5 begin
1391:C      5 readsize := sourceize - 1*pagesize; { scs 1/17/83 }
1392:C      5 if readsize > pagesize then readsize := pagesize; { scs 1/17/83 }
1393:C      5 readblocks(filelib.fbp^, textbuf.p^, readsize, { scs 1/17/83 }
1394:C      5 ptr := textbuf; linestart := true; fileblock+sourceblock+1*pageblocks);
1395:C      5 repeat
1396:C      6 case ptr.cp^ of
1397:C      6 chr(etc),
1398:C      6 nulchr: ptr.a := textbuf.a + pagesize;
1399:C      6 otherwise dochar(ptr.cp^);
1400:C      6 end;
1401:C      6 ptr.a := ptr.a + 1;
1402:C      6 until ptr.a >= textbuf.a + pagesize;
1403:C      5 if not linestart then dochar(eol);
1404:C      5 end;
1405:C      4 end;
1406:C      3 listln;
1407:C      3 newmods := modsave;
1408:C      3 lowheap := infost;
1409:C      3 end;
1410:C
1411:S
1412:D      2 procedure disassemble;
1413:D      3 begin
1414:D      3 prepunassem;
1415:D      3 nilgvr := Nil;
1416:D      3 getcodeblocks;
1417:D      3 newmods := modsave;
1418:D      3 lowheap := infost;
1419:D      3 end;
1420:S
1421:D      2 procedure getbounds;
1422:D      3 begin
1423:D      3 lastline := -1;
1424:D      3 fgotoxy(output, 0,13);
1425:D      3 write('lower bound? ');
1426:D      3 if readint(lowrange) then
1427:D      4 begin
1428:D      4 write('upper bound? ');
1429:D      4 if not readint(highrange) then
1430:D      5 highrange := maxint;
1431:D      4 end;
1432:D      4 else begin
1433:D      4 lowrange := minint;
1434:D      4 highrange := maxint;
1435:D      4 end;
1436:D      3 end;
1437:S
1438:C      2 begin {unassemble}
1439:C      2 fortranflag := false;
1440:C      2 decodestate := notype; dumped := false;

```



```

1441:C      2  repeat
1442:C      3    gotoxy(output, 0,2);
1443:C      3    writeln('Q Quit', cteos);
1444:C      3    writeln('S Stop unassembling');
1445:C      3    writeln('I print Import table');
1446:C      3    writeln('E print Ext table');
1447:C      3    writeln('D print Def table');
1448:C      3    writeln('A unassemble all (Assembler conventions)');
1449:C      3    writeln('C unassemble all (Compiler conventions)');
1450:C      3    writeln('P PC range (Assembler conventions)');
1451:C      3    writeln('L Line range (Compiler conventions)', cteos);
1452:C      3    getcommandchar('unassemble option?', commandchar);
1453:C      3    if commandchar <> ' ' then
1454:C      4      case commandchar of
1455:C      5        'S': decodestate := quittype;
1456:C      5        'Q': begin
1457:C      5          decodestate := quittype;
1458:C      5          quit;
1459:C      5          end;
1460:C      5        'A': begin rangetype := norange; decodestate := abscode; disassemble; end;
1461:C      5        'C': begin rangetype := norange; decodestate := const; disassemble; end;
1462:C      5        'P': begin rangetype := pcrange; decodestate := abscode; getbounds; disassemble; end;
1463:C      5        'L': begin rangetype := linerange; decodestate := const; getbounds; disassemble; end;
1464:C      5        'I': listtext;
1465:C      5        'D': listdefs;
1466:C      5        'E': listexts;
1467:C      5        otherwise do beep;
1468:C      5        end;
1469:C      3    until decodestate = quittype;
1470:C      2 end; {unassemble}
1471:S
1472:S
1473:D      1 procedure makenewgvr(var oldptr: addrec;
1474:D      2                          modptr: moddescptr);
1475:S
1476:D      -2 2 var refsize: shortint;
1477:D      -2 2 lastptr;
1478:D      -2 2 firstptr;
1479:D      -2 2 vptr;
1480:D      -18 2 gptr: addrec;
1481:S
1482:D      2 procedure runlist(var oldptr: addrec; modptr: moddescptr; sub: boolean);
1483:D      -1 3 var done: boolean;
1484:D      -6 3 defptr: addrec;
1485:S
1486:D      3 procedure addref(add: shortint; sub: boolean);
1487:D      -12 4 var iptr, jptr, tp: addrec;
1488:D      -12 4 notdone;
1489:D      -14 4 notcancels: boolean;
1490:C      4 begin
1491:C      4   if add = 0 then
1492:C      5     if sub then vptr.vcp^.value := vptr.vcp^.value - modptr^.relocdelta
1493:C      6     else vptr.vcp^.value := vptr.vcp^.value + modptr^.relocdelta
1494:C      6   else if add = 1 then
1495:C      6     if sub then vptr.vcp^.value := vptr.vcp^.value - modptr^.globaldelta
1496:C      7     else vptr.vcp^.value := vptr.vcp^.value + modptr^.globaldelta;
1497:S
1498:C      4   iptr := lastptr; notdone := true; notcancels := true;
1499:C      4   while (iptr.a > firstptr.a) and notdone do
1500:C      5     begin

```

```

1501:C      5   iptr.a := iptr.a - sizeof(referenceptr);
1502:C      5   with iptr.rpp^ do
1503:C      6     if adr <= add then
1504:C      7       begin
1505:C      7         if adr = add then notcancels := (op = subit) = sub;
1506:C      7         iptr.a := iptr.a + sizeof(referenceptr);
1507:C      7         notdone := false;
1508:C      7         end;
1509:C      5   end;
1510:S
1511:C      4   if notcancels then
1512:C      5     begin
1513:C      5       gbytes(jptr.a, sizeof(referenceptr));
1514:C      5       lastptr := lowheap;
1515:C      5       while jptr.a > iptr.a do
1516:C      6         begin
1517:C      6           tp.a := jptr.a - sizeof(referenceptr);
1518:C      6           jptr.rpp^ := tp.rpp^;
1519:C      6           jptr := tp;
1520:C      6           end;
1521:C      5       with iptr.rpp^ do
1522:C      6         begin adr := add; last := false;
1523:C      6         if sub then op := subit else op := addit;
1524:C      6         end;
1525:C      5       end
1526:C      5     else
1527:C      5       begin
1528:C      5         tp.a := iptr.a - sizeof(referenceptr);
1529:C      5         while iptr.a < lastptr.a do
1530:C      6           begin
1531:C      6             tp.rpp^ := iptr.rpp^;
1532:C      6             tp := iptr;
1533:C      6             iptr.a := iptr.a + sizeof(referenceptr);
1534:C      6             end;
1535:C      5         lastptr.a := lastptr.a - sizeof(referenceptr);
1536:C      5         lowheap := lastptr;
1537:C      5         end;
1538:C      4   end;
1539:S
1540:C      3   begin {runlist}
1541:C      3   with oldptr.gvp^ do
1542:C      4     begin
1543:C      4       if longoffset then oldptr.a := oldptr.a+sizeof(generalvalue, true)
1544:C      5       else oldptr.a := oldptr.a+sizeof(generalvalue, false);
1545:C      4       if valueextend then
1546:C      5         begin
1547:C      5           if sub then vptr.vcp^.value := vptr.vcp^.value - oldptr.vcp^.value
1548:C      6           else vptr.vcp^.value := vptr.vcp^.value + oldptr.vcp^.value;
1549:C      5           oldptr.a := oldptr.a + sizeof(valueextension, sint);
1550:C      5           end;
1551:C      4       if primarytype <> absolute then
1552:C      5         begin
1553:C      5           if modptr = NIL then
1554:C      6             begin
1555:C      6               modptr := newmods; done := false;
1556:C      6               repeat
1557:C      7                 with modptr^ do
1558:C      8                   if patchmod then modptr := link
1559:C      9                   else if oldptr.a < defaddr.a then modptr := link
1560:C      10                  else if oldptr.a > defaddr.a + defsize then modptr := link

```



```

1561:C      11      else done := true;
1562:C      7      until done;
1563:C      6      end;
1564:C      6      case primarytype of
1565:C      6      relocate:      addr(0, sub);
1566:C      6      global:      addr(1, sub);
1567:C      6      general:
1568:C      6      begin
1569:C      6      done := false;
1570:C      6      repeat with oldptr.rpp^ do
1571:C      6      begin
1572:C      6      defptr := modptr^.extaddr.ptp^[adr];
1573:C      6      if modptr^.unresbits.bmp^[adr] then
1574:C      6      addr(defptr.rp.adr, sub <> (op = subit))
1575:C      9      else
1576:C      9      begin
1577:C      9      defptr.a := defptr.a + strlen(defptr.syp^)+ 2
1578:C      9      - ord(odd(strlen(defptr.syp^)));
1579:C      9      runlist(defptr, NIL, sub <> (op = subit));
1580:C      9      end;
1581:C      6      oldptr.a := oldptr.a + sizeof(referenceptr);
1582:C      6      done := last;
1583:C      6      end;
1584:C      7      until done;
1585:C      6      end; (general)
1586:C      6      end; (case)
1587:C      6      end; (primarytype <> absolute)
1588:C      4      end; (with)
1589:C      3      end; (runlist)
1590:C
1591:C      2      begin (makenewgvr)
1592:C      2      gbytes(gptr.a, sizeof(generalvalue));
1593:C      2      gptr.gvp^ := oldptr.gvp^;
1594:C      2      with gptr.gvp^ do
1595:C      3      begin
1596:C      3      if not longoffset then
1597:C      4      lowheap.a := lowheap.a -
1598:C      4      (sizeof(generalvalue) - sizeof(generalvalue, false));
1599:C      3      gbytes(vptr.a, sizeof(valueextension, sint));
1600:C      3      vptr.vep^.value := 0;
1601:C      3      valueextend := true;
1602:C      3      end;
1603:C
1604:C      2      firstptr := lowheap; lastptr := firstptr;
1605:C
1606:C      2      runlist(oldptr, modptr, false);
1607:C      2      with gptr.gvp^ do
1608:C      3      begin
1609:C      3      refsize := lastptr.a - firstptr.a;
1610:C      3      if refsize = 0 then primarytype := absolute
1611:C      4      else
1612:C      4      begin
1613:C      4      if refsize = sizeof(referenceptr) then with firstptr.rpp^ do
1614:C      6      if adr <= 1 then if op = addit then
1615:C      8      begin
1616:C      8      if adr = 0 then primarytype := relocate
1617:C      8      else primarytype := global;
1618:C      8      lastptr := firstptr;
1619:C      8      lowheap := lastptr;
1620:C      8      refsize := 0;

```

```

1621:C      8      end;
1622:C      4      if refsize > 0 then
1623:C      5      begin
1624:C      5      firstptr.a := lastptr.a - sizeof(referenceptr);
1625:C      5      firstptr.rpp^.last := true;
1626:C      5      end;
1627:C      3      end;
1628:C      3      short := lastptr.a - gptr.a;      (even if it is long variety)
1629:C      3      end;
1630:C      2      end;
1631:C
1632:C
1633:C      1      procedure compressgvr(gvptr: addrec);
1634:C      2      var vptr: addrec;
1635:C      2      begin
1636:C      2      with gvptr.gvp^ do if valueextend then
1637:C      4      begin
1638:C      4      if longoffset then vptr.a := gvptr.a + sizeof(generalvalue, true);
1639:C      4      else vptr.a := gvptr.a + sizeof(generalvalue, false);
1640:C      4      with vptr.vep^ do
1641:C      5      if value = 0 then
1642:C      6      begin
1643:C      6      lowheap.a := lowheap.a - sizeof(valueextension, sint);
1644:C      6      fastmove(point(vptr.a + sizeof(valueextension, sint)), vptr.p,
1645:C      6      lowheap.a - vptr.a);
1646:C      6      valueextend := false; short := short - sizeof(valueextension, sint);
1647:C      6      end;
1648:C      4      end;
1649:C      2      end;
1650:C
1651:C      1      procedure rsolve;
1652:C      -12      2      var modptr, lastptr, nextptr: moddescptr;
1653:C      -20      2      mrbase, mgbase: integer;
1654:C      -24      2      sp: addrec;
1655:C      -26      2      len: shortint;
1656:C      -28      2      i: shortint;
1657:C
1658:C      2      begin
1659:C      2      modptr := newmods; lastptr := NIL; (reverse the pointers)
1660:C      2      while modptr <> NIL do with modptr^ do
1661:C      4      begin nextptr := link; link := lastptr;
1662:C      4      lastptr := modptr; modptr := nextptr;
1663:C      4      end;
1664:C      2      newmods := lastptr;
1665:C
1666:C      2      startgvr.p := NIL; startgvr.mod := NIL;
1667:C      2      modptr := newmods; totalpatchspace := 0;
1668:C      2      forwardpatches:=NIL; backwardpatches:=NIL;
1669:C      2      mrbase := startreloc; mgbase := startglobal;
1670:C      2      while modptr <> NIL do with modptr^ do
1671:C      4      begin
1672:C      4      if patchmod then
1673:C      5      begin
1674:C      5      patchbase := mrbase;
1675:C      5      mrbase := mrbase + patchsize;
1676:C      5      totalpatchspace := totalpatchspace + patchsize;
1677:C      5      if forwardpatches = NIL then forwardpatches := modptr
1678:C      5      else lastptr^.patchlink := modptr;
1679:C      5      lastptr := modptr;
1680:C      5      end

```



```

1681:C      5      else with directory.drp^ do
1682:C      6      begin
1683:C      6          relocbase := mrbase; relocdelta := mrbase - relocatablebase;
1684:C      6          mrbase := mrbase + relocatablebase + ord(odd(relocatablebase));
1685:C      6          globase := mgbase; globaldelta := mgbase - globalbase;
1686:C      6          mgbase := mgbase - globalbase - ord(odd(globalbase));
1687:C      6
1688:C      6          gbytes(unresbits.a, ((extsize div 4 + 15) div 16)*2);
1689:C      6          for i := 2 to extsize div 4 - 1 do unresbits.bmp[i] := false;
1690:C      6          unresbits.bmp[0] := true; unresbits.bmp[1] := true;
1691:C      6          extaddr.ptp[0].rp.w := 0;
1692:C      6          extaddr.ptp[1].rp.w := 4;
1693:C      6
1694:C      6          sp := directory;
1695:C      6          sp.a := sp.a + sizeof(moduledirectory);
1696:C      6
1697:C      6          if newmodname.syp = NIL then newmodname := sp;
1698:C      6          if startgvr.p = NIL then
1699:C      7              if executable then
1700:C      8                  begin
1701:C      8                      startgvr.mod := modptr;
1702:C      8                      startgvr.a := sp.a + strlen(sp.syp^)*2 - ord(odd(strlen(sp.syp^)));
1703:C      8                  end;
1704:C      6          end;
1705:C      4          modptr := link;
1706:C      4          end;
1707:C      2          totalreloc := mrbase - startreloc;
1708:C      2          totalglobal := startglobal - mgbase;
1709:C      2
1710:C      2      end; {rsolve}
1711:C
1712:C
1713:C      1      procedure mergeexts;
1714:C
1715:C      -4      2      var alist:      address;
1716:C      -8      2      slist:      sortlistptr;
1717:C      -12     2      sptr:      address;
1718:C      -14     2      listlen:   shortint;
1719:C      -16     2      sortlen:   shortint;
1720:C      -18     2      minindex:  shortint;
1721:C      -22     2      modptr:   moddescptr;
1722:C      -22     2      len,
1723:C      -30     2      i:      integer;
1724:C      -31     2      done:   boolean;
1725:C      -31     2      strptr,
1726:C      -40     2      newstrptr: address;
1727:C
1728:C      2      begin
1729:C      2          slist := lowheap.slp; listlen := 0;
1730:C      2          modptr := newmods;
1731:C      2          while modptr <> NIL do with modptr^ do
1732:C      4              begin
1733:C      4                  if not patchmod then if not resolved then
1734:C      6                      begin
1735:C      6                          listlen := listlen + 1;
1736:C      6                          gbytes(sptr.a, sizeof(sortdesc));
1737:C      6                          with sptr.sdp^ do
1738:C      7                              begin
1739:C      7                                  modp := modptr;
1740:C      7                                  N := 0;

```

```

1741:C      7          end;
1742:C      6          end;
1743:C      4          modptr := link;
1744:C      4          end;
1745:C
1746:C      2          gbytes(list.a, listlen * sizeof(shortint));
1747:C      2          for i := 1 to listlen do ilist.ilp[i] := i-1;
1748:C      2          sortlen := listlen;
1749:C      2          gbytes(newexttable, 8);
1750:C      2          newextsize := 8;
1751:C
1752:C      2          while listlen > 0 do
1753:C      3              begin
1754:C      3                  while sortlen > 0 do with slist[ilist.ilp[sortlen]], modp^ do
1755:C      5                      begin
1756:C      5                          done := false;
1757:C      5                          repeat
1758:C      6                              if N >= listsize then
1759:C      7                                  begin
1760:C      7                                      listlen := listlen - 1;
1761:C      7                                      for i := sortlen to listlen do ilist.ilp[i] := ilist.ilp[i+1];
1762:C      7                                      done := true;
1763:C      7                                      end;
1764:C      7                                      else if listaddr[N] = 0 then N := N + 1
1765:C      8                                      else
1766:C      8                                          begin
1767:C      8                                              ext := symbolptr(extaddr.a + listaddr[N]);
1768:C      8                                              i := sortlen; minindex := ilist.ilp[i];
1769:C      8                                              repeat
1770:C      9                                                  if i >= listlen then done := true
1771:C      10                                                 else if ext^ <= slist[ilist.ilp[i+1]].ext^ then done := true
1772:C      11                                                 else begin ilist.ilp[i] := ilist.ilp[i+1]; i := i + 1; end;
1773:C      9                                                  until done;
1774:C      8                                                  ilist.ilp[i] := minindex;
1775:C      8                                                  end;
1776:C      6                                                  until done;
1777:C      6                                                  sortlen := sortlen - 1;
1778:C      5                                                  end;
1779:C      3              end;
1780:C      4              if listlen > 0 then
1781:C      4                  begin
1782:C      4                      strptr.syp := slist[ilist.ilp[1]].ext;
1783:C      4                      len := strlen(strptr.syp) + 4 - strlen(strptr.syp) mod 4;
1784:C      4                      gbytes(newstrptr.a, len);
1785:C      4                      fastmove(strptr.p, newstrptr.p, len);
1786:C      4                      i := 1; done := false;
1787:C      6                      repeat with slist[ilist.ilp[i]], modp^ do
1788:C      7                          if ext^ = newstrptr.syp^ then
1789:C      7                              begin
1790:C      7                                  wordrecptr(ext)^.w := newextsize;
1791:C      7                                  unresbits.bmp[listaddr[N] div 4] := true;
1792:C      7                                  N := N + 1;
1793:C      7                                  i := i + 1; done := i > listlen;
1794:C      7                                  end;
1795:C      5                      else done := true;
1796:C      4                      until done;
1797:C      4                      sortlen := 1-1;
1798:C      4                      newextsize := newextsize + len;
1799:C      4                      end;
1800:C      3                  end;

```



```

1801:C      2   if newextsize <= 8 then newextsize := 0;
1802:S
1803:C      2 end;
1804:
1805:      1 function gvrequa(a,b: address; offset: integer): boolean;
1806:D      2 var      boff: integer;
1807:D      -8      aoff: integer;
1808:D      -12     b0:   gvrptr;
1809:C      2 begin
1810:C      2 gvrequa := false;
1811:C      2 b0 := b.gvp;
1812:C      2 with a.gvp do
1813:C      2   if primarytype = b0.primarytype then
1814:C      2     begin
1815:C      2       if longoffset then a.a := a.a + 4;
1816:C      2       else a.a := a.a + 2;
1817:C      2       if b0.longoffset then b.a := b.a + 4;
1818:C      2       else b.a := b.a + 2;
1819:C      2       if valueextend then
1820:C      2         begin
1821:C      2           aoff := a.vcp.value;
1822:C      2           a.a := a.a + sizeof(valueextension, sint);
1823:C      2         end;
1824:C      2       else aoff := 0;
1825:C      2       if b0.valueextend then
1826:C      2         begin
1827:C      2           boff := b.vcp.value;
1828:C      2           b.a := b.a + sizeof(valueextension, sint);
1829:C      2         end;
1830:C      2       else boff := 0;
1831:C      2       if aoff + offset = boff then
1832:C      2         if primarytype = general then
1833:C      2           begin
1834:C      2             while (a.rpp.w
1835:C      2               = b.rpp.w)
1836:C      2               and (a.rpp.last = false) do
1837:C      2               begin
1838:C      2                 a.a := a.a + sizeof(referenceptr);
1839:C      2                 b.a := b.a + sizeof(referenceptr);
1840:C      2               end;
1841:C      2               gvrequa := a.rpp.w = b.rpp.w;
1842:C      2             end;
1843:C      2             else gvrequa := true;
1844:C      2           end;
1845:C      2       end;
1846:S
1847:D      1 procedure mkdir;
1848:S
1849:D      -4   2 var modptr:      moddescptr;
1850:D      -4   2 newtextrec;
1851:D      -12  2 lasttextrec:  address;
1852:D      -14  2 len:          shortint;
1853:D      -14  2 extblocks,
1854:D      -14  2 newtextrecs,
1855:D      -14  2 movebytes,
1856:D      -30  2 textrecs:      integer;
1857:D      -30  2 index,
1858:D      -38  2 newptr:      address;
1859:D      -38  2 tempdirptr,
1860:D      -38  2 oldindex,

```

```

1861:D      -38  2 oldptr,
1862:D      -54  2 ptr:          address;
1863:S
1864:S
1865:D      2 procedure mergetext;
1866:D      -1   2 var merged: boolean;
1867:D      -1   2 lastptr,
1868:D      -10  2 newptr:  address;
1869:C      2 begin
1870:C      2   if newtextrec.tdp.textsize = 0 then lowheap := newtextrec
1871:C      2   else
1872:C      2     begin
1873:C      2       if lasttextrec.tdp <> NIL then
1874:C      2         begin
1875:C      2           lastptr.a := lasttextrec.a + sizeof(textdescriptor);
1876:C      2           newptr.a := newtextrec.a + sizeof(textdescriptor);
1877:C      2           merged := gvrequa(lastptr, newptr, lasttextrec.tdp.textsize);
1878:C      2         end;
1879:C      2         else merged := false;
1880:S
1881:C      2         if merged then
1882:C      2           begin
1883:C      2             lasttextrec.tdp.textsize := lasttextrec.tdp.textsize + newtextrec.tdp.textsize;
1884:C      2             lowheap := newtextrec;
1885:C      2           end;
1886:C      2         else
1887:C      2           begin newtextrecs := newtextrecs + 1;
1888:C      2             lasttextrec := newtextrec;
1889:C      2           end;
1890:C      2         end;
1891:C      2       end;
1892:S
1893:C      2 begin {mkdir}
1894:C      2   gbytes(tempdirptr.a, sizeof(moduledirectory));
1895:C      2   if newmodname.syp=NIL then
1896:C      2     begin gbytes(newmodname.a, 2);
1897:C      2       newmodname.syp := ' ';
1898:C      2     end;
1899:C      2   else
1900:C      2     begin
1901:C      2       len := strlen(newmodname.syp) + 2 - ord(odd(strlen(newmodname.syp)));
1902:C      2       gbytes(index, len);
1903:C      2       fastmove(newmodname.p, point(index), len);
1904:C      2     end;
1905:C      2   if startgvr.p<>NIL then
1906:C      2     begin
1907:C      2       oldptr := startgvr;
1908:C      2       ptr := lowheap;
1909:C      2       makenewgvr(oldptr, startgvrmod);
1910:C      2       compressgvr(ptr);
1911:C      2     end;
1912:C      2   lasttextrec.tdp := NIL;
1913:C      2   newtextrecs := 0;
1914:C      2   modptr := newmods;
1915:C      2   while modptr <> NIL do with modptr do
1916:C      2     begin
1917:C      2       if patchmod then
1918:C      2         begin
1919:C      2           gbytes(newtextrec.a, sizeof(textdescriptor));
1920:C      2         end;

```



```

1921:C      5      newtextrec.tdp^.textsize := patchsize;
1922:C      5      gbytes(olddptr.a, sizeof(generalvalue, false));
1923:C      5      with oldptr.gvp do
1924:C      6      begin
1925:C      6      primarytype := relocatable;      datasize := sint;
1926:C      6      patchable := false;      longoffset := false;
1927:C      6      if patchbase = 0 then begin valueextend := false; short := 2; end
1928:C      7      else begin
1929:C      7      gbytes(newptr, sizeof(valueextension, sint));
1930:C      7      veptr(newptr).value := patchbase;
1931:C      7      valueextend := true; short := 6;
1932:C      7      end
1933:C      7      end;
1934:C      5      mergetext;
1935:C      5      end
1936:C      5      else with directory.drp^ do
1937:C      6      begin
1938:C      6      oldindex.a := directory.a + sizeof(moduledirectory);
1939:C      6      oldindex.a := oldindex.a + strlen(oldindex.syp^)+2 -
1940:C      6      ord(odd(strlen(oldindex.syp^)));
1941:C      6      if executable then oldindex.a := oldindex.a + oldindex.gvp^.short;
1942:C      6      textrecs := textrecords;
1943:C      6      while textrecs > 0 do with oldindex.tdp^ do
1944:C      8      begin
1945:C      8      gbytes(newtextrec.a, sizeof(textdescriptor));
1946:C      8      if odd(textsize) then textsize := textsize + 1;
1947:C      8      newtextrec.tdp^.textsize := textsize;
1948:C      8      oldindex.a := oldindex.a + sizeof(textdescriptor);
1949:C      8      ptr := lowheap; makenewgvr(oldindex, modptr);
1950:C      8      compressgvr(ptr);
1951:C      8      mergetext;
1952:C      8      textrecs := textrecs - 1;
1953:C      8      end;
1954:C      6      end;
1955:C      4      modptr := link;
1956:C      4      end;
1957:C      2      with tempdirptr.drp^ do
1958:C      3      begin
1959:C      3      date := todaydate;
1960:C      3      revision := linkerdate;
1961:C      3      producer := 'L';
1962:C      3      systemid := 3;
1963:C      3      notice := copyright;
1964:C      3      directorysize := lowheap.a - tempdirptr.a;
1965:C      3      (modulesize := )
1966:C      3      {executable := (startgvr.p <> NIL);
1967:C      3      relocatablebase := totalreloc;
1968:C      3      relocatablebase := startreloc;
1969:C      3      globalsize := totalglobal;
1970:C      3      globalbase := startglobal;
1971:C      3      {extblock := }
1972:C      3      {extsize := }
1973:C      3      {defblock := }
1974:C      3      {defsize := }
1975:C      3      sourceblock := 0;      {implement later}
1976:C      3      sourceblock := 0;
1977:C      3      sourceblock := 0;
1978:C      3      textrecords := newtextrecs;
1979:C      3      nextblock := (directorysize + (blocksize-1)) div blocksize;
1980:C      3

```

```

1981:S      3      extblock := nextblock;
1982:C      3      extsize := newextsize;
1983:C      3      extblocks := (newextsize + (blocksize-1)) div blocksize;
1984:C      3      blockwrite(outfile, point(newexttable)^, extblocks, outblock+nextblock);
1985:C      3      nextblock := nextblock + extblocks;
1986:C      3
1987:S      3
1988:S      3      {lowheap.a := newdirectory.a + directorysize;
1989:S      3      fastmove(tempdirptr.p, newdirectory.p, directorysize);
1990:S      3      newdirectory := tempdirptr;
1991:S      3      end;
1992:S      3
1993:C      2      end;
1994:S      1
1995:D      1      procedure mergedefs;
1996:S      2
1997:D      -4      2      var slist:      sortlistptr;
1998:D      -8      2      list:      address;
1999:D      -10      2      listlen:      shortint;
2000:D      -12      2      sortlen:      shortint;
2001:D      -14      2      minindex:      shortint;
2002:D      -18      2      modptr:      moddescptr;
2003:D      -18      2      len:      integer;
2004:D      -20      2      i:      integer;
2005:D      -27      2      done:      boolean;
2006:D      -27      2      strptr:      address;
2007:D      -27      2      newstrptr:      address;
2008:D      -40      2      sptr:      address;
2009:S      2
2010:D      -44      2      newdeftable: address;
2011:D      -48      2      defblocks: integer;
2012:D      -49      2      c:      char;
2013:S      2
2014:C      2      begin
2015:C      2      slist := lowheap.slp; listlen := 0;
2016:C      2      modptr := newmods;
2017:C      2      while modptr <> NIL do with modptr^ do
2018:C      4      begin
2019:C      4      if not patchmod then if defsize > 0 then
2020:C      6      begin
2021:C      6      listlen := listlen + 1;
2022:C      6      gbytes(sptr.a, sizeof(sortdesc));
2023:C      6      with sptr.sdp^ do
2024:C      7      begin
2025:C      7      modp := modptr;
2026:C      7      def := defaddr;
2027:C      7      end;
2028:C      6      end;
2029:C      4      modptr := link;
2030:C      4      end;
2031:S      2
2032:C      2      gbytes(list.a, listlen * sizeof(shortint));
2033:C      2      for i := 1 to listlen do list.ilp[i] := i-1;
2034:C      2      sortlen := listlen;
2035:C      2      newdeftable := lowheap.a;
2036:S      2
2037:C      2      while listlen > 0 do
2038:C      3      begin
2039:C      3      while sortlen > 0 do with slist^[list.ilp[sortlen]], modp^ do
2040:C      5      begin

```



```

2041:C      5      done := false;
2042:C      5      repeat
2043:C      6      if def.a >= defaddr.a + defsize then
2044:C      7      begin
2045:C      7      listlen := listlen - 1;
2046:C      7      for i := sortlen to listlen do ilist.ilp[i] := ilist.ilp[i+1];
2047:C      7      done := true;
2048:C      7      end
2049:C      7      else
2050:C      7      begin
2051:C      7      len := strlen(def.syp^)+2 - ord(odd(strlen(def.syp^)));
2052:C      7      with gvrptr(def.a+len) do
2053:C      8      if patchable then def.a := def.a + len + short
2054:C      9      else
2055:C      9      begin
2056:C      9      i := sortlen; minindex := ilist.ilp[i];
2057:C      9      repeat
2058:C      10     if i >= listlen then done := true
2059:C      11     else if def.syp^ <= slist^[ilist.ilp[i+1]].def.syp^ then done := true
2060:C      12     else begin ilist.ilp[i] := ilist.ilp[i+1]; i := i + 1; end;
2061:C      10     until done;
2062:C      9     ilist.ilp[i] := minindex;
2063:C      9     end;
2064:C      7     end
2065:C      7     until done;
2066:C      5     sortlen := sortlen - 1;
2067:C      5     end;
2068:C      3     if listlen > 0 then
2069:C      4     begin
2070:C      4     with slist^[ilist.ilp[1]] do
2071:C      5     begin
2072:C      5     strptr := def;
2073:C      5     len := strlen(strptr.syp^)+2 - ord(odd(strlen(strptr.syp^)));
2074:C      5     gbytes(newstrptr.a, len);
2075:C      5     fastmove(strptr.p, newstrptr.p, len);
2076:C      5     def.a := strptr.a + len;
2077:C      5     makenewgvr(def, modp);
2078:C      5     end;
2079:C      4     i := 2; done := false;
2080:C      4     repeat with slist^[ilist.ilp[i]], modp^ do
2081:C      6     if i > listlen then done := true
2082:C      7     else if def.syp^ = newstrptr.syp^ then
2083:C      8     begin
2084:C      8     if printer on then
2085:C      9     begin
2086:C      9     list; writeln(listing, 'duplicate symbol definition for: ',
2087:C      9     def.syp^); (**!!*)
2088:C      9     end
2089:C      8     else
2090:C      8     begin
2091:C      8     errorline;
2092:C      8     writeln('duplicate symbol: ', def.syp^);
2093:C      8     if streaming then escape(119);
2094:C      8     write('Press ''C'' to continue, any other key to abort ', cteol);
2095:C      8     read(keyboard, c);
2096:C      8     if (c <> 'C') and (c <> 'c') then escape(119);
2097:C      8     fgotoxy(output, 0, 22);
2098:C      8     writeln(cteol);
2099:C      8     write('LINKING ...', cteol);
2100:C      8     end;

```

```

2101:C      3      def.a := def.a + strlen(def.syp^)+2 - ord(odd(strlen(def.syp^)));
2102:C      3      def.a := def.a + def.gvp^.short;
2103:C      3      i := i + 1;
2104:C      3      end
2105:C      3      else done := true;
2106:C      3      until done;
2107:C      4      sortlen := 1-1;
2108:C      4      end;
2109:C      3      end;
2110:S
2111:C      2      with newdirectory.drp^ do
2112:C      3      begin
2113:C      3      defblock := nextblock;
2114:C      3      if defsize then defsize := lowheap.a - newdeftable
2115:C      4      else defsize := 0;
2116:C      3      defblocks := (defsize + (blocksize-1)) div blocksize;
2117:C      3      if defblocks > 0 then
2118:C      4      blockwrite(outfile, point(newdeftable)^, defblocks, outblock+nextblock);
2119:C      3      nextblock := nextblock + defblocks;
2120:C      3      end;
2121:C      2      lowheap.slp := slist;
2122:S
2123:C      2      end;
2124:S
2125:D      1      procedure copytext;
2126:S
2127:D      -4      2      var patchptr: patchdescptr;
2128:D      -12      loadaddr, loadaddr0: address;
2129:D      -16      modptr: moddescptr;      {current module being loaded}
2130:D      -20      gvrptr;
2131:D      -20      patching;
2132:D      -22      merging:      boolean;      {whether text records are combined}
2133:S
2134:D      -22      2      textbuffer,      {base of text record buffer}
2135:D      -22      textbufftop,      {end of text record buffer}
2136:D      -22      textindex,      {pointer to next space available in text buffer}
2137:D      -22      object,      {object in text record being modified by ref record}
2138:S
2139:D      -22      2      refbuffer,      {base of ref table buffer}
2140:D      -22      refbufftop,      {end of ref table buffer}
2141:D      -22      outrefindex,      {pointer to next space available in ref buffer}
2142:D      -22      inrefindex,      {pointer to next record in ref buffer to process}
2143:S
2144:D      -22      2      newptr,      {base of new gvr on heap}
2145:D      -22      valptr,      {value extension in new gvr on heap}
2146:S
2147:D      -22      2      oldindex,      {pointer to old text descriptors}
2148:D      -22      newindex:      {pointer to new text descriptors}
2149:D      -70      2      address;
2150:S
2151:D      -70      2      vvalue;
2152:D      -70      2      newbytes,      {size of new gvr on heap}
2153:D      -70      2      offsetbytes,      {distance from last object referenced by new refs}
2154:D      -70      2      oldtextrec,      {text records left to process from old module}
2155:S
2156:D      -70      2      textbuffblocks,      {maximum blocks allocated for text buffer}
2157:D      -70      2      textinblock,      {file relative block index into old text}
2158:D      -70      2      textinsize,      {number of bytes left to read from old text}
2159:D      -70      2      textoutblock,      {file relative block index into new text}
2160:D      -70      2      textoutsize,      {number of bytes processed into new text}

```



```

2161:S
2162:D -70 2   refbufblocks,      (maximum blocks allocated for ref buffer)
2163:D -70 2   refinblock,      (file relative block index into old ref table)
2164:D -70 2   refsize,        (number of bytes left to read from old ref)
2165:D -70 2   refoutblock,    (file relative block index into new ref table)
2166:D -70 2   refoutsize,     (number of bytes processed into new ref table)
2167:D -126 2 integer;
2168:S
2169:D
2170:D
2171:C
2172:C
2173:C
2174:C
2175:C
2176:S
2177:C
2178:C
2179:C
2180:S
2181:C
2182:C
2183:C
2184:C
2185:C
2186:C
2187:C
2188:C
2189:C
2190:C
2191:C
2192:C
2193:S
2194:D
2195:D -4
2196:D -8
2197:D -12
2198:C
2199:C
2200:C
2201:C
2202:C
2203:C
2204:C
2205:C
2206:C
2207:C
2208:C
2209:C
2210:C
2211:C
2212:C
2213:C
2214:C
2215:C
2216:C
2217:C
2218:C
2219:C
2220:C

```

```

procedure starttext;
begin
  if not merging then with newindex.tdp do
  begin
    textstart := nextblock;      textoutblock := nextblock + outblock;
    nextblock := nextblock + (textsize + (blocksize - 1)) div blocksize;
    refoutblock := nextblock + outblock;
  end;
  textoutsize := 0;              textindex := textbuffer;
  refoutsize := 0;              outrefindex := refbuffer;
  offsetbytes := 0;             object := textbuffer;
  valptr.a := newindex.a + sizeof(textdescriptor);
  patching := (valptr.gvp^.primarytype = relocatable) and (totalpatchspace > 0);
  if patching then
  begin
    if valptr.gvp^.valueextend then
    begin
      valptr.a := valptr.a + sizeof(generalvalue, false);
      loadaddr0 := valptr.vvp^.value;
    end;
  end;
  else loadaddr0 := 0;
  loadaddr := object.a - loadaddr0;
end;
end;

procedure endtext;
var lastblocks: integer;
td: address;
org: integer;
begin
  with newindex.tdp do
  begin
    merging := (textoutsize < textsize);
    if not merging then
    begin
      if textindex.a > textbuffer.a then
      begin
        lastblocks := (textindex.a - textbuffer.a + (blocksize - 1)) div blocksize;
        blockwrite(outfile, textbuffer.p, lastblocks, textoutblock);
      end;
      if outrefindex.a > refbuffer.a then
      begin
        lastblocks := (outrefindex.a - refbuffer.a + (blocksize - 1)) div blocksize;
        blockwrite(outfile, refbuffer.p, lastblocks, refoutblock);
      end;
      refstart := nextblock;      refsize := refoutsize;
      nextblock := nextblock + (refoutsize + (blocksize - 1)) div blocksize;
      newindex.a := newindex.a + sizeof(textdescriptor);
      org := 0; gvrstring(newindex.gvp, org, false, true);
      if printer on then
      begin
        list; writeln(listing);
      end;
    end;
  end;
end;

```

```

2221:C
2222:C
2223:C
2224:C
2225:C
2226:S
2227:D
2228:D -4
2229:C
2230:C
2231:C
2232:C
2233:C
2234:C
2235:C
2236:C
2237:C
2238:C
2239:C
2240:S
2241:D
2242:D -8
2243:C
2244:C
2245:C
2246:C
2247:C
2248:C
2249:C
2250:C
2251:C
2252:C
2253:C
2254:C
2255:C
2256:C
2257:C
2258:C
2259:C
2260:C
2261:C
2262:C
2263:C
2264:C
2265:S
2266:D
2267:D -8
2268:C
2269:C
2270:C
2271:C
2272:C
2273:C
2274:C
2275:C
2276:C
2277:C
2278:S
2279:D
2280:D

```

```

'load record: size = ', textsize:1, ', load address = ', gvaluestring, '');
end;
end;
end;
end;

procedure dumptext(writebytes: integer);
var writeblocks: integer;
begin
  writeblocks := writebytes div blocksize;
  writebytes := writeblocks * blocksize;
  blockwrite(outfile, textbuffer.p, writeblocks, textoutblock);
  textoutblock := textoutblock + writeblocks;
  textindex.a := textindex.a + writebytes;
  object.a := object.a + writebytes;
  loadaddr := loadaddr + writebytes;
  fastmove(point(textbuffer.a + writebytes), textbuffer.p,
    textindex.a - textbuffer.a);
end;

procedure checktextbuf(observe: integer);
var readbytes, writebytes: integer;
begin
  while textindex.a < object.a + observe do
  begin
    readbytes := textbuftop.a - textindex.a;
    if textinsize <= readbytes then readbytes := textinsize
  end;
  if object.a < textindex.a then writebytes := object.a - textbuffer.a
  else writebytes := textindex.a - textbuffer.a;
  if writebytes < readbytes then
    readbytes := readbytes - readbytes mod blocksize
  else begin dumptext(writebytes); readbytes := 0; end;
  end;
  if readbytes > 0 then
  begin
    readblocks(modptr^.filefb.fbp, textindex.p, readbytes, textinblock);
    textinblock := textinblock + readbytes div blocksize;
    textinsize := textinsize - readbytes;
    textindex.a := textindex.a + readbytes;
  end;
end;

procedure dumprefs;
var writebytes, writeblocks: integer;
begin
  writeblocks := (outrefindex.a - refbuffer.a) div blocksize;
  writebytes := writeblocks * blocksize;
  blockwrite(outfile, refbuffer.p, writeblocks, refoutblock);
  refoutblock := refoutblock + writeblocks;
  outrefindex.a := outrefindex.a + writebytes;
  fastmove(point(refbuffer.a + writebytes), refbuffer.p,
    outrefindex.a - refbuffer.a);
end;

procedure checkinref;
const maxrefsize = 254;

```



```
2281:D -8 3 var refindex, readbytes: integer;
2282:C 3 begin
2283:C 3   refindex := refbuftop.a - inrefindex.a;
2284:C 3   if refindex < maxrefsize then
2285:C 4     begin
2286:C 4       if refindex > 0 then
2287:C 5         repeat
2288:C 6           if outrefindex.a > refbuffer.a + blocksize then
2289:C 7             readbytes := inrefindex.a - {outrefindex.a + blocksize}
2290:C 7           else readbytes := inrefindex.a - {refbuffer.a + (2 * blocksize)};
2291:C 6           if refindex <= readbytes then readbytes := refindex
2292:C 7           else if outrefindex.a - refbuffer.a < readbytes then
2293:C 8             readbytes := readbytes - readbytes mod blocksize
2294:C 8           else begin dumprefs; readbytes := 0; end;
2295:C 6           if readbytes > 0 then
2296:C 7             begin
2297:C 7               fastmove(inrefindex.p, point(inrefindex.a - readbytes), refindex);
2298:C 7               inrefindex.a := inrefindex.a - readbytes;
2299:C 7               readblocks(modptr^.filefib.fbp, point(inrefindex.a + refindex)^, readbytes, refindex);
2300:C 7               refindex := refindex + readbytes div blocksize;
2301:C 7               refindex := refindex - readbytes;
2302:C 7             end;
2303:C 7           until readbytes > 0;
2304:C 6         end;
2305:C 4       end;
2306:C 3     end;
2307:S
2308:D 2 procedure putref;
2309:D -2 3 var newbytes: shortint;
2310:D -6 3 valptr: address;
2311:C 3 begin
2312:C 3   compressgvr(newptr);
2313:C 3   with newptr.gvp do
2314:C 4     if longoffset then long := offsetbytes
2315:C 5     else if offsetbytes < 256 then short := offsetbytes
2316:C 6     else
2317:C 6       begin
2318:C 6         valptr.a := newptr.a + sizeof(generalvalue, false);
2319:C 6         moveright(valptr.p, point(valptr.a + 2)^,
2320:C 6           lowheap.a - valptr.a);
2321:C 6         lowheap.a := lowheap.a + 2;
2322:C 6         longoffset := true;
2323:C 6         long := offsetbytes;
2324:C 6       end;
2325:C 3   offsetbytes := 0;
2326:C 3   newbytes := lowheap.a - newptr.a;
2327:C 3   if outrefindex.a + newbytes > inrefindex.a then dumprefs;
2328:C 3   fastmove(newptr.p, outrefindex.p, newbytes);
2329:C 3   outrefindex.a := outrefindex.a + newbytes;
2330:C 3   refoutsize := refoutsize + newbytes;
2331:C 3   lowheap := newptr;
2332:C 3   end; {putref}
2333:S
2334:D 2 procedure patcherror(dsize: datatype);
2335:S
2336:D 3 procedure printmessage(var f: text);
2337:D -4 4 var index: address;
2338:C 4 begin
2339:C 4   index.a := modptr^.directory.a + sizeof(moduledirectory);
2340:C 4   write(f, 'Can't patch byte ';
```

```
2341:C 4   object.a - loadaddr - loadaddr0:1,
2342:C 4   ' in text record ', oldtextrec:1,
2343:C 4   ' of module ', index.syp^);
2344:C 4   end;
2345:S
2346:D 3 begin
2347:C 3   errors := errors + 1;
2348:C 3   errorline; printmessage(output);
2349:C 3   if printer on then
2350:C 4     begin list;
2351:C 4       write(listing, '*** ERROR *** ');
2352:C 4       printmessage(listing); writeln(listing);
2353:C 4     end
2354:C 3   else escape(128);
2355:C 3   end;
2356:S
2357:D 2 procedure makepatch;
2358:D -8 3 var r, rptr: address;
2359:D -12 3 objectaddr: address;
2360:D -16 3 patchaddr: address;
2361:D -24 3 foundpatchmodptr:
2362:D -24 3 patchmodptr: moddescptr;
2363:D -24 3 foundlastpptr:
2364:D -24 3 lastpatchptr:
2365:D -36 3 patchptr: patchdescptr;
2366:D -36 3 patchdelta:
2367:D -48 3 delta2, foundpatchdelta: integer;
2368:D -50 3 patchstate: (nopatch, longpatch, shortpatch, oldpatch);
2369:D -51 3 backwardlist: boolean;
2370:C 3 begin
2371:C 3   objectaddr := object.a - loadaddr;
2372:C 3   with valptr.vcp do
2373:C 4     value := value + object.sw^ + objectaddr;
2374:C 3   with newptr.gvp do
2375:C 4     begin
2376:C 4       patchable := false;
2377:C 4       if primarytype = absolute then primarytype := relocatable
2378:C 5       else
2379:C 5         begin
2380:C 5           if primarytype <> general then
2381:C 6             begin
2382:C 6               gbytes(rptr.a, sizeof(referenceptr));
2383:C 6               with rptr.rpp do
2384:C 7                 begin adr := 0; op := addit; last := false; end;
2385:C 6               gbytes(rptr.a, sizeof(referenceptr));
2386:C 6               with rptr.rpp do
2387:C 7                 begin adr := ord(primarytype)-1;
2388:C 7                 op := addit; last := true; end;
2389:C 6               primarytype := general; short := short + 4;
2390:C 6             end
2391:C 5           else
2392:C 6             begin
2393:C 6               rptr.a := valptr.a + sizeof(valueextension, sint);
2394:C 6               with rptr.rpp do
2395:C 7                 if (adr=0) and (op=subit) then
2396:C 8                   if last then
2397:C 9                     begin
2398:C 9                       primarytype := absolute; lowheap := rptr;
2399:C 9                       short := short - 2;
2400:C 9                     end
```



```

2401:C      9      else
2402:C      9      begin
2403:C      9      moveleft(point(rptr.a+2)^,rptr.p^,
2404:C      9      short-(valptr.a-newptr.a)-6);
2405:C      9      lowheap.a := lowheap.a - sizeof(referenceptr);
2406:C      9      short := short - 2;
2407:C      9      end
2408:C      9      else
2409:C      8      begin
2410:C      8      gbytes(r.a, sizeof(referenceptr));
2411:C      8      moveright(rptr.p^,point(rptr.a+2)^,
2412:C      8      short-(valptr.a-newptr.a)-4);
2413:C      8      adr := 0; op := addit; last := false;
2414:C      8      short := short + 2;
2415:C      8      end;
2416:C      6      end;
2417:C      5      end;
2418:C      4      end;
2419:C      3      patchstate := nopatch;
2420:C      3      for backwardlist := false to true do
2421:C      4      begin
2422:C      4      if backwardlist then patchmodptr := backwardpatches
2423:C      5      else patchmodptr := forwardpatches;
2424:C      4      while (patchmodptr <> NIL) and (patchstate < oldpatch) do
2425:C      5      with patchmodptr^ do
2426:C      6      begin
2427:C      6      patchaddr := patchbase;
2428:C      6      patchptr := patchlist;
2429:C      6      while (patchptr <> NIL) and (patchstate < oldpatch) do
2430:C      7      with patchptr^ do
2431:C      8      begin
2432:C      8      patchdelta := patchaddr-objectaddr;
2433:C      8      if (-32768<=patchdelta) and (patchdelta<32768)
2434:C      9      then if gvrequal(newptr,patchref,0)
2435:C      10     then begin
2436:C      10     object.sw^ := patchdelta;
2437:C      10     lowheap := newptr;
2438:C      10     patchstate := oldpatch;
2439:C      10     end;
2440:C      8     if patchref.gvp^.datasize = sword then
2441:C      9     patchaddr := patchaddr + 4;
2442:C      9     else patchaddr := patchaddr + 6;
2443:C      8     lastpatchptr := patchptr;
2444:C      8     patchptr := patchlist;
2445:C      8     end;
2446:C      6     patchdelta := patchaddr-objectaddr;
2447:C      6     if (patchstate < shortpatch) then
2448:C      7     if (-32768<=patchdelta) and (patchdelta<32768) then
2449:C      8     begin
2450:C      8     if patchsize - (patchaddr - patchbase) >= 4 then
2451:C      9     if newptr.gvp^.primarytype = relocatable then
2452:C      10    begin
2453:C      10    delta2 := valptr.vcp^.value - (patchaddr+2);
2454:C      10    if (-32768 <= delta2) and (delta2 < 32768) then
2455:C      11    if object.a + patchdelta >= textbuffer.a then
2456:C      12    begin
2457:C      12    patchstate := shortpatch;
2458:C      12    foundpatchdelta := patchdelta;
2459:C      12    foundlastpptr := lastpatchptr;
2460:C      12    foundpatchmodptr := patchmodptr;

```

```

2461:C      12     end;
2462:C      12     end;
2463:C      8     if (patchstate < longpatch) and not backwardlist then
2464:C      9     if patchsize - (patchaddr - patchbase) >= 6 then
2465:C      10    begin
2466:C      10    patchstate := longpatch;
2467:C      10    foundpatchdelta := patchdelta;
2468:C      10    foundlastpptr := lastpatchptr;
2469:C      10    foundpatchmodptr := patchmodptr;
2470:C      10    end;
2471:C      8     end;
2472:C      6     patchmodptr := patchlink;
2473:C      6     end;
2474:C      4     end;
2475:C      3     if patchstate = nopatch then patcherror(newptr.gvp^.datasize)
2476:C      4     else if patchstate < oldpatch then
2477:C      5     with foundpatchmodptr^ do
2478:C      6     begin
2479:C      6     gbytes(r.a, sizeof(patchdescriptor));
2480:C      6     if patchlist = NIL then patchlist := r.pdp
2481:C      7     else foundlastpptr^.patchlist := r.pdp;
2482:C      6     with r.pdp^ do
2483:C      7     begin
2484:C      7     patchlist := NIL;
2485:C      7     patchref := newptr;
2486:C      7     if patchstate = longpatch then newptr.gvp^.datasize := sint
2487:C      8     else begin
2488:C      8     newptr.gvp^.datasize := sword;
2489:C      8     if foundpatchdelta < 0 then
2490:C      9     begin
2491:C      9     r.a := object.a + foundpatchdelta;
2492:C      9     r.sw.w := 24576 {BRA pc relative};
2493:C      9     r.a := r.a + 2;
2494:C      9     r.sw^ := delta2;
2495:C      9     if printer on then
2496:C      10    begin
2497:C      10    list; write(listing, '(backward patch) BRA ');
2498:C      10    gvrp := patchref.gvp; vvalue := 0;
2499:C      10    gvrstring(gvrp, vvalue, false, true);
2500:C      10    writeln(listing, gvaluestring,
2501:C      10    '':20-strlen(gvaluestring),
2502:C      10    r.a-2-loadaddr:10);
2503:C      10    end;
2504:C      9    end;
2505:C      8    end;
2506:C      7    end;
2507:C      6    object.sw^ := foundpatchdelta;
2508:C      6    end;
2509:C      3    end;
2510:C      2    begin {procedure copytext}
2511:C      2    {estimate data structures at 3/2(totalpatchspace) + 1/4(workspace) }
2512:C      2    textbufblocks := ((highheap.a - lowheap.a) * 3 -
2513:C      2    totalpatchspace * 6) div (blocksize * 4);
2514:C      2
2515:C      2    refbufblocks := textbufblocks div 4;
2516:C      2    if refbufblocks < 4 then refbufblocks := 4;
2517:C      2
2518:C      2    textbufblocks := textbufblocks - refbufblocks;
2519:C      2    if textbufblocks < 3 then textbufblocks := 3;
2520:C      2

```



```
2521:C      2 gbytes(textbuffer.a, textbufblocks * blocksize);
2522:C      2 textbuftop := lowheap;
2523:C      2
2524:C      2 gbytes(refbuffer.a, refbufblocks * blocksize);
2525:C      2 refbuftop := lowheap;
2526:C      2
2527:C      2 newindex.a := newdirectory.a + sizeof(moduledirectory);
2528:C      2 newindex.a := newindex.a + strlen(newindex.syp^) + 2 -
2529:C      2      ord(odd(strlen(newindex.syp^)));
2530:C      2 if newdirectory.drp^.executable then
2531:C      2     newindex.a := newindex.a + newindex.gvp^.short;
2532:C      2
2533:C      2 merging := false;
2534:C      2 modptr := newmods;
2535:C      2 while modptr <> NIL do with modptr^ do
2536:C      4     begin
2537:C      4         if patchmod then
2538:C      5             begin
2539:C      5                 if printeron then
2540:C      6                     begin
2541:C      6                         list; writeln(listing, '(patch space)', patchsize:29, patchbase:10);
2542:C      6                     end;
2543:C      6                     starttext;
2544:C      6                     patchptr := patchlist;
2545:C      6                     while patchptr <> NIL do with patchptr^, patchref.gvp^ do
2546:C      7                         begin
2547:C      7                             if textbuftop.a - textindex.a < 6 then
2548:C      8                                 dumptext(textindex.a - textbuffer.a);
2549:C      7                             if printeron then
2550:C      8                                 begin list;
2551:C      8                                     gvrp := patchref.gvp; vvalue := 0;
2552:C      8                                     gvrstring(gvrp, vvalue, false, true);
2553:C      8                                     end;
2554:C      8                                     if valueextend then
2555:C      8                                         begin
2556:C      8                                             if longoffset then valptr.a := patchref.a + sizeof(generalvalue, true)
2557:C      9                                             else valptr.a := patchref.a + sizeof(generalvalue, false);
2558:C      8                                             vvalue := valptr.vcp^.value;
2559:C      8                                             end;
2560:C      8                                         else vvalue := 0;
2561:C      7                                         if datasize = sword (PC relative branch) then
2562:C      8                                             begin
2563:C      8                                                 if printeron then
2564:C      9                                                     writeln(listing, ' BRA ', gvaluestring,
2565:C      9                                                         '':26-strlen(gvaluestring), object.a-loadaddr:20);
2566:C      8                                                 object.uw^.w := 24576 (BRA PC relative);
2567:C      8                                                 object.a := object.a + 2;
2568:C      8                                                 object.su^ := vvalue - (object.a - loadaddr);
2569:C      8                                                 object.a := object.a + 2;
2570:C      8                                                 offsetbytes := offsetbytes + 4;
2571:C      8                                                 end;
2572:C      8                                             else (long absolute branch)
2573:C      8                                                 begin
2574:C      8                                                     if printeron then
2575:C      9                                                         writeln(listing, ' JMP ', gvaluestring,
2576:C      9                                                             '':26-strlen(gvaluestring), object.a-loadaddr:20);
2577:C      8                                                         object.uw^.w := 20217 (JMP long absolute);
2578:C      8                                                         object.a := object.a + 2;
2579:C      8                                                         object.si^ := vvalue;
2580:C      8                                                         object.a := object.a + 4;
```

```
2581:C      8 gbytes(newptr.a, short);
2582:C      8 fastmove(patchref.p, newptr.p, short);
2583:C      8 if valueextend then
2584:C      9     begin
2585:C      9         valptr.a := newptr.a + (valptr.a - patchref.a);
2586:C      9         valptr.vcp^.value := 0;
2587:C      9     end;
2588:C      8 offsetbytes := offsetbytes + 2;
2589:C      8 putref;
2590:C      8 offsetbytes := 4;
2591:C      8 end;
2592:C      7 textindex := object;
2593:C      7 patchptr := patchlist;
2594:C      7 end;
2595:C      5 object.a := textindex.a + patchsize - (object.a - loadaddr - patchbase);
2596:C      5 while textindex.a < object.a do
2597:C      6     begin
2598:C      6         if textindex.a >= textbuftop.a - 2 then
2599:C      7             dumptext(textindex.a - textbuffer.a);
2600:C      6             textindex.sw^ := -1;
2601:C      6             textindex.a := textindex.a + 2;
2602:C      6             offsetbytes := offsetbytes + 2;
2603:C      6         end;
2604:C      5 textoutsize := textoutsize + patchsize;
2605:C      5 endtext;
2606:C      5 forwardpatches := patchlink;
2607:C      5 patchlink := backwardpatches;
2608:C      5 backwardpatches := modptr;
2609:C      5 end;
2610:C      5 else with directory.drp^ do
2611:C      6     begin
2612:C      6         oldindex.a := directory.a + sizeof(moduledirectory);
2613:C      6         if printeron then
2614:C      7             begin
2615:C      7                 list; writeln(listing, oldindex.syp^, '':32-strlen(oldindex.syp^),
2616:C      7                     relocatable:10, relocbase:10,
2617:C      7                     global:10, globalbase:10);
2618:C      7                 end;
2619:C      6         oldindex.a := oldindex.a + strlen(oldindex.syp^)+2 -
2620:C      6             ord(odd(strlen(oldindex.syp^)));
2621:C      6         if executable then oldindex.a := oldindex.a + oldindex.gvp^.short;
2622:C      6         for oldtextrec := 1 to textrecords do
2623:C      7             begin
2624:C      7                 if oldindex.tdp^.textsize > 0 then
2625:C      8                     begin
2626:C      8                         starttext;
2627:C      8                         loadaddr0 := object.a - loadaddr;
2628:C      8                         with oldindex.tdp^ do
2629:C      9                             begin
2630:C      9                                 refinsize := refsize;
2631:C      9                                 textinsize := textsize;
2632:C      9                                 refinblock := fileblock+refstart;
2633:C      9                                 textinblock := fileblock+textstart;
2634:C      9                                 textoutsize := textoutsize + textsize;
2635:C      9                             end;
2636:C      8                     end;
2637:C      8                     inrefindex := refbuftop;
2638:C      8                     while (refbuftop.a - inrefindex.a) + refinsize > 0 do
2639:C      9                         begin
```



```

2641:C      9      checkinref;
2642:C      9      newptr := lowheap;
2643:C      9      with inrefindex.gvp^ do
2644:C      10      if longoffset then
2645:C      11      begin
2646:C      11      newbytes := long;
2647:C      11      valptr.a := newptr.a + sizeof(generalvalue, true);
2648:C      11      end
2649:C      11      else
2650:C      11      begin
2651:C      11      newbytes := short;
2652:C      11      valptr.a := newptr.a + sizeof(generalvalue, false);
2653:C      11      end;
2654:C      9      object.a := object.a + newbytes;
2655:C      9      offsetbytes := offsetbytes + newbytes;
2656:C      9      makenewgvr(inrefindex, modptr);
2657:C      9      with newptr.gvp^, valptr.vep^ do
2658:C      10      begin
2659:C      10      case datasize of ($range offs)
2660:C      11      sbyte:
2661:C      11      begin
2662:C      11      checktextbuf(sizeof(sbyterec));
2663:C      11      value := value + object.sb^.sb;
2664:C      11      object.sb^.sb := value;
2665:C      11      value := value - object.sb^.sb;
2666:C      11      end;
2667:C      11      sword:
2668:C      11      begin
2669:C      11      checktextbuf(sizeof(shortint));
2670:C      11      value := value + object.sw^;
2671:C      11      object.sw^ := value;
2672:C      11      value := value - object.sw^;
2673:C      11      end;
2674:C      11      sint:
2675:C      11      begin
2676:C      11      checktextbuf(sizeof(integer));
2677:C      11      object.si^ := object.si^ + value;
2678:C      11      value := 0;
2679:C      11      end;
2680:C      11      ubyte:
2681:C      11      begin
2682:C      11      checktextbuf(sizeof(ubyterec));
2683:C      11      value := value + object.ub^.ub;
2684:C      11      object.ub^.ub := value;
2685:C      11      value := value - object.ub^.ub;
2686:C      11      end;
2687:C      11      uword:
2688:C      11      begin
2689:C      11      checktextbuf(sizeof(wordrec));
2690:C      11      value := value + object.uw^.w;
2691:C      11      object.uw^.w := value;
2692:C      11      value := value - object.uw^.w;
2693:C      11      end; ($range on$)
2694:C      11      otherwise escape(111);
2695:C      11      end; (case datasize)
2696:C      10      if primarytype = absolute then
2697:C      11      begin
2698:C      11      if value <> 0 then
2699:C      12      if patchable and patching then makepatch
2700:C      13      else patcherror(datasize);

```

```

2701:C      11      end
2702:C      11      else if patching and patchable then makepatch
2703:C      12      else putref;
2704:C      10      end; (with gvrptr(newptr)^, valptr^)
2705:C      9      end; (while there are any refs)
2706:C      S
2707:C      8      newbytes := textindex.a + textinsize - object.a;
2708:C      8      offsetbytes := offsetbytes + newbytes;
2709:C      8      object.a := object.a + newbytes;
2710:C      8      checktextbuf(0);
2711:C      S
2712:C      8      endtext;
2713:C      8      end;
2714:C      7      oldindex.a := oldindex.a + sizeof(textdescriptor);
2715:C      7      oldindex.a := oldindex.a + oldindex.gvp^.short;
2716:C      7      end; (for oldtextrec)
2717:C      6      end; (with directory^ do)
2718:C      4      modptr := link;
2719:C      4      end;
2720:C      2      end; (copytext)
2721:C      S
2722:C      1      procedure printdirectentry(modnum: shortint; var entry: direntry);
2723:C      2      begin
2724:C      2      with entry do
2725:C      3      begin
2726:C      3      upc(dtid);
2727:C      3      list, write(listing, modnum:4, ' ', dtid,
2728:C      3      dlastblk-dfirstblk:21-strlen(dtid), ' ');
2729:C      3      writedata(listing, daccess);
2730:C      3      writeln(listing, dfirstblk:7);
2731:C      3      end;
2732:C      2      end;
2733:C      S
2734:C      1      procedure bootmod(modnum: shortint);
2735:C      2      const sectorsize = 256;
2736:C      2      var buffer, bufptr, valptr, ptr, endrefs, mname,
-28      infostart:
2737:C      2      object, recordnum: integer;
-36
2738:C      2
2739:C      S
2740:C      2      procedure writesector(anyvar f: fib; anyvar obj: window; size, sector: integer);
2741:C      3      begin
2742:C      3      call (f.am, addr(f), writebytes, obj, size, sector * sectorsize);
2743:C      3      if ioresult <> 0 then escape(114);
2744:C      3      end;
2745:C      S
2746:C      2      begin
2747:C      2      infostart := lowheap;
2748:C      2      loadinfo(modnum, true, true);
2749:C      2      with newmods^, directory.drp^ do
2750:C      3      begin
2751:C      3      if extsize > 8 then escape(120);
2752:C      3      mname.a := directory.a + sizeof(moduledirectory);
2753:C      3      ptr.a := mname.a + strlen(mname.syp^)+2 - ord(odd(strlen(mname.syp^)));
2754:C      3      if executable then with ptr.gvp^, fibp(addr(outfile))^ do
2755:C      5      begin
2756:C      5      if fstartaddress = 0 then
2757:C      6      if valueextend then
2758:C      7      begin
2759:C      7      valptr.a := ptr.a + sizeof(generalvalue, false);
2760:C      7      fstartaddress := valptr.vep^.value;

```



```
2761:C      7      end;
2762:C      5      ptr.a := ptr.a + short;
2763:C      5      end;
2764:C      3      recordnum := 0;
2765:C      3      while textrecords > 0 do with ptr.tdp^ do
2766:C      3      begin
2767:C      3          recordnum := recordnum + 1;
2768:C      3          if refsize > 0 then (check to make sure code is "absolute")
2769:C      3              begin
2770:C      3                  gbytes(buffer.a, refsize);
2771:C      3                  readblocks(filefib.fbp^, buffer.p^, refsize, fileblock + refstart);
2772:C      3                  bufptr := buffer; endrefs.a := buffer.a + refsize;
2773:C      3                  object := 0;
2774:C      3                  while bufptr.a < endrefs.a do with bufptr.gvp^ do
2775:C      3                      begin
2776:C      3                          if longoffset then
2777:C      3                              begin object := object + long;
2778:C      3                                  bufptr.a := bufptr.a + sizeof(generalvalue, true);
2779:C      3                              end
2780:C      3                          else begin object := object + short;
2781:C      3                                  bufptr.a := bufptr.a + sizeof(generalvalue, false);
2782:C      3                              end;
2783:C      3                          if valueextend then
2784:C      3                              begin
2785:C      3                                  errorline;
2786:C      3                                  write ('Can't relocate byte ', object:1,
2787:C      3                                      ' in record ', recordnum:1,
2788:C      3                                      ' of module ', mname.syp^);
2789:C      3                                  escape(128);
2790:C      3                              end;
2791:C      3                          if primarytype = general then
2792:C      3                              begin
2793:C      3                                  while not bufptr.rpp^.last do
2794:C      3                                      bufptr.a := bufptr.a + sizeof(referenceptr);
2795:C      3                                      bufptr.a := bufptr.a + sizeof(referenceptr);
2796:C      3                                  end;
2797:C      3                              end;
2798:C      3                              lowheap := buffer;
2799:C      3                              end;
2800:C      3                              gbytes(buffer.a, sizeof(integer));
2801:C      3                              ptr.a := ptr.a + sizeof(textdescriptor);
2802:C      3                              with ptr.gvp^ do
2803:C      3                                  begin
2804:C      3                                      if valueextend then
2805:C      3                                          begin
2806:C      3                                              valptr.a := ptr.a + sizeof(generalvalue, false);
2807:C      3                                              buffer.p^ := valptr.vgp^.value;
2808:C      3                                          end
2809:C      3                                      else buffer.p^ := 0;
2810:C      3                                      ptr.a := ptr.a + short;
2811:C      3                                      end;
2812:C      3                              gbytes(bufptr.a, sizeof(integer));
2813:C      3                              bufptr.p^ := textsize;
2814:C      3                              end;
2815:C      3                              gbytes(bufptr.a, textsize);
2816:C      3                              readblocks(filefib.fbp^, bufptr.p^, textsize, fileblock + textstart);
2817:C      3                              writesector(outfile, buffer.p^, textsize + 2 * sizeof(integer), outblock);
2818:C      3                              outblock := outblock +
2819:C      3                                  (textsize + 2 * sizeof(integer) + (sectorsize - 1)) div sectorsize;
2820:C      3                              end;
```

```
2821:C      5      lowheap := buffer;
2822:C      5      textrecords := textrecords - 1;
2823:C      5      end;
2824:C      3      end;
2825:C      2      lowheap := infostart;
2826:C      2      newmods := NIL;
2827:C      2      end;
2828:C      1      procedure copymodule(modnum: shortint);
2829:C      1      const fsize = sizeof(addr) * sizeof(fib, 1);
2830:C      1      fsize := fsize + odd(odd(fsize));
2831:C      1      var startblock, numblocks, transblocks: shortint;
2832:C      -6      copybuffer: addr;
2833:C      -10      bufblocks: shortint;
2834:C      -12      begin
2835:C      2      if booting then bootmod(modnum)
2836:C      2      else if linking then begin
2837:C      4          if loadfib.a >= highheap.a then
2838:C      4              begin
2839:C      5                  fastmove(highheap.p, lowheap.p, fsize);
2840:C      5                  highheap.a := highheap.a + fsize;
2841:C      5                  lowheap.a := lowheap.a + fsize;
2842:C      5                  loadfib.a := loadfib.a - (highheap.a - lowheap.a);
2843:C      5                  end;
2844:C      4                  loadinfo(modnum, true, true)
2845:C      4                  end
2846:C      4      else
2847:C      4          begin
2848:C      4              bufblocks := (highheap.a - lowheap.a) div blocksize;
2849:C      4              gbytes(copybuffer.a, bufblocks * blocksize);
2850:C      4              if outmodnum >= maxmodules then escape(127);
2851:C      4              outmodnum := outmodnum + 1;
2852:C      4              outdirectory.fdp^[outmodnum] := fdirectory^[modnum];
2853:C      4              with fdirectory^[modnum] do
2854:C      5                  begin
2855:C      5                      startblock := dfirstblk; numblocks := dlastblk - startblock;
2856:C      5                      end;
2857:C      5                      with outdirectory.fdp^[outmodnum] do
2858:C      5                          begin
2859:C      5                              dfirstblk := outblock; dlastblk := outblock + numblocks;
2860:C      5                              end;
2861:C      5                              while numblocks > 0 do
2862:C      5                                  begin
2863:C      5                                      if numblocks <= bufblocks then transblocks := numblocks
2864:C      5                                          else transblocks := bufblocks;
2865:C      5                                      readblocks(loadfib.fbp^, copybuffer.p^, transblocks * blocksize, startblock);
2866:C      5                                      blockwrite(outfile, copybuffer.p^, transblocks, outblock);
2867:C      5                                      startblock := startblock + transblocks;
2868:C      5                                      outblock := outblock + transblocks;
2869:C      5                                      numblocks := numblocks - transblocks;
2870:C      5                                      end;
2871:C      5                                      lowheap := copybuffer;
2872:C      5                                      (if printer on then printdirectentry(outmodnum, outdirectory.fdp^[outmodnum]));
2873:C      5                                  end;
2874:C      5                              end;
2875:C      4          end;
2876:C      2      end;
2877:C      1      procedure writedirectory;
2878:C      2      begin
2879:C      2          with newdirectory.drp^ do
```



```

2881:C      3 begin
2882:C      3 modulesize := nextblock * blocksize;
2883:C      3 blockwrite(outfile, newdirectory.drp^, extblock, outblock);
2884:C      3 end;
2885:S
2886:C      2 if outmodnum >= maxmodules then escape(127);
2887:C      2 outmodnum := outmodnum + 1;
2888:C      2 with outdirectory.fdp[outmodnum] do
2889:C      2 begin
2890:C      3 dfirstblk := outblock;
2891:C      3 outblock := outblock + nextblock;
2892:C      3 dlastblk := outblock;
2893:C      3 dfkind := codefile;
2894:C      3 moveleft(newmodname.syp^, dtid, sizeof(filename));
2895:C      3 if strlen(dtid) > flength then setstrlen(dtid, flength);
2896:C      3 dlastbyte := 256;
2897:C      3 daccess := todaysdate;
2898:C      3 end;
2899:C      2 if printeron then printdirectentry(outmodnum, outdirectory.fdp[outmodnum]); }
2900:C      2 end;
2901:S
2902:D      1 procedure trim(var s: string);
2903:D      2 var first, last: shortint;
2904:D      2 begin
2905:D      3 last := strlen(s);
2906:D      3 while last > 0 do
2907:D      4 begin
2908:D      5 if s[last] = ' ' then
2909:D      6 begin last := last - 1; if last = 0 then s := ''; end
2910:D      7 else
2911:D      8 begin
2912:D      9 first := 1; while s[first] = ' ' do first := first + 1;
2913:D      9 s := str(s, first, last - first + 1); last := 0;
2914:D      9 end;
2915:D      4 end;
2916:D      2 end;
2917:S
2918:D      1 procedure toggleprinter;
2919:D      2 var newlistname: string80;
2920:D      2 fvid: vid;
2921:D      2 ftitle: fid;
2922:D      2 fsegs: integer;
2923:D      2 fkind: filekind;
2924:C      2 begin
2925:C      2 printeron := not printeron;
2926:C      2 if printeron then
2927:C      3 begin
2928:C      4 fgotoxy(output, 13,3); write(' ', cteol);
2929:C      4 readln(newlistname);
2930:C      4 fixname(newlistname, textfile);
2931:C      4 if scantitle(newlistname, fvid, ftitle, fsegs, fkind) then ; { jws 3/2/84 }
2932:C      4 if strlen(newlistname) > 0 then
2933:C      5 begin
2934:C      6 listfilename := newlistname;
2935:C      6 if fsegs = 0 then { jws 3/2/84 }
2936:C      7 sappend(newlistname, '[*]');
2937:C      6 pageeject;
2938:C      6 if (pagenum = 0) and (linenum = 0)
2939:C      7 then close(listing)
2940:C      7 else close(listing, 'lock');

```

```

2941:C      4 rewrite(listing, newlistname);
2942:C      4 pagenum := 0; linenum := 0;
2943:C      4 printopen := loresult = 0;
2944:C      4 printeron := printopen;
2945:C      4 if not printopen then escape(118);
2946:C      4 end;
2947:C      4 else printeron := printopen;
2948:C      3 end;
2949:C      2 end;
2950:S
2951:D      1 procedure copyon;
2952:D      2 begin
2953:D      2 lowheap := infostart;
2954:D      2 linking := false;
2955:D      2 end;
2956:S
2957:D      1 procedure closein;
2958:D      2 begin
2959:D      2 if fdirectory <> NIL then
2960:D      3 begin
2961:D      4 if loadfib.a >= highheap.a then
2962:D      5 begin
2963:D      6 close(loadfib.php^);
2964:D      6 loadfib.a := loadfib.a - sizeof(addrac);
2965:D      6 loadfib := loadfib.arpa;
2966:D      6 end;
2967:D      4 highheap := highheap0; fdirectory := NIL;
2968:D      4 vmodnum := 0; verifying := false;
2969:D      3 end;
2970:D      2 end;
2971:S
2972:D      1 procedure initlink;
2973:D      2 begin
2974:D      2 linking := true; defsout := true;
2975:D      2 infostart := lowheap; newmodname.syp := NIL;
2976:D      2 startreloc := 0; startglobal := 0;
2977:D      2 copyright := '';
2978:D      2 end;
2979:S
2980:D      1 procedure link;
2981:D      2 begin
2982:D      2 errors := 0;
2983:D      2 fgotoxy(output, 0,23); write('LINKING ...');
2984:D      2 rsolve;
2985:D      2 if printeron then
2986:D      3 begin
2987:D      4 list; writeln(listing, 'link map', 'Rsize':34, 'Rbase':10, 'Gsize':10, 'Gbase':10);
2988:D      4 list; writeln(listing, '-----':42, '-----':10, '-----':10, '-----':10);
2989:D      4 end;
2990:D      2 newdirectory := lowheap;
2991:D      2 mergeexts;
2992:D      2 makedir; {also write new ext table, move down directory}
2993:D      2 mergedefs;
2994:D      2 copytext;
2995:D      2 writedirectory;
2996:S
2997:C      2 if printeron then
2998:C      3 begin
2999:C      4 list; writeln(listing, '-----':42, '-----':20);
3000:C      4 list;

```



```

3001:C      3   if newmodname.syp = NIL then write(listing, '(no name)', '32-9)
3002:C      4   else write(listing, newmodname.syp, '32-strlen(newmodname.syp));
3003:C      3   writeln(listing, totalreloc:10, totalglobal:20);
3004:C      3   listln;
3005:C      3   end;
3006:C      2   closein;
3007:C      2   closefiles;
3008:C      2   lowheap := infostart; (release memory used by linker)
3009:C      2   linking := false;   newmods := NIL;
3010:C      2   if errors > 0 then escape(122);
3011:C      2   end; (link)
3012:S
3013:S      1 procedure printdirectory;
3014:D      2 var numfiles: shortint;
3015:D      -2 2 modnum: shortint;
3016:D      -4 2
3017:C      2 begin
3018:C      2 list; writeln(listing, 'FILE DIRECTORY OF: ', loadfib.fbid, '');
3019:C      2 listln;
3020:C      2 numfiles := fdirectory^[0].dnumfiles;
3021:C      2 for modnum := 1 to numfiles do
3022:C      3   printdirectentry(modnum, fdirectory^[modnum]);
3023:C      2 listln;
3024:C      2 end;
3025:S
3026:D      1 procedure copyfile;
3027:D      2 var modnum: shortint;
3028:C      -2 2
3029:C      2 begin
3030:C      2   for modnum := 1 to fdirectory^[0].dnumfiles do
3031:C      3   copymodule(modnum);
3032:C      2   closein;
3033:C      2   end;
3034:D      1 procedure verifynext;
3035:C      2 begin
3036:C      2   if vmodnum < fdirectory^[0].dnumfiles then
3037:C      3   begin
3038:C      3   vmodnum := vmodnum + 1;
3039:C      3   upc(fdirectory^[vmodnum].dtid)
3040:C      3   end
3041:C      3   else
3042:C      3   begin
3043:C      3   vmodnum := 0;
3044:C      3   verifying := false;
3045:C      3   end;
3046:C      2   end;
3047:S
3048:D      1 procedure verifymod;
3049:C      2 begin
3050:C      2   vmodnum := 0; verifying := true;
3051:C      2   verifynext;
3052:C      2   end;
3053:S
3054:D      1 procedure xfer;
3055:D      -2 2 var modnum: shortint;
3056:C      2 begin
3057:C      2   if verifying then
3058:C      3   begin
3059:C      3   copymodule(vmodnum);
3060:C      3   verifynext;

```

```

3061:C      3   end
3062:C      3   else
3063:C      3   begin
3064:C      3   for modnum := 1 to fdirectory^[0].dnumfiles do
3065:C      4   with fdirectory^[modnum] do
3066:C      5   begin
3067:C      5   upc(dtid);
3068:C      5   if dtid = fdirectory^[vmodnum].dtid then
3069:C      5   copymodule(modnum);
3070:C      5   end;
3071:C      3   vmodnum := 0;
3072:C      3   end;
3073:C      2   end;
3074:S
3075:D      1 procedure openin;
3076:D      2 const fsize = sizeof(addr) + sizeof(fib, 1);
3077:D      2 fsize = fsize + odd(fsize);
3078:C      2 begin
3079:C      2   closein;
3080:C      2   fgotoxy(output, 22, 13); write(cteol);
3081:C      2   if strlen(infile) = 0 then (if no name then get it)
3082:C      3   begin readln(infile); fixname(infile, codefile);
3083:C      3   end;
3084:C      2   if strlen(infile) > 0 then
3085:C      3   begin
3086:C      3   openlinkfile(infile);
3087:C      3   if fdirectory = NIL then
3088:C      4   begin errorline;
3089:C      4   write('cannot open ', infile, ' ');
3090:C      4   ioerror;
3091:C      4   end
3092:C      4   else
3093:C      4   begin highheap.a := highheap.a - fsize;
3094:C      4   lowheap.a := lowheap.a - fsize;
3095:C      4   fastmove(lowheap.p, highheap.p, fsize);
3096:C      4   loadfib.a := loadfib.a + (highheap.a - lowheap.a);
3097:C      4   if fdirectory^[0].dnumfiles = 1 then vmodnum := 1
3098:C      5   else verifymod;
3099:C      5   end;
3100:C      2   end;
3101:C      2   end; (openin)
3102:S
3103:D      1 procedure closeout;
3104:C      2 begin
3105:C      2   closein;
3106:C      2   with outdirectory.fdp^[0] do
3107:C      3   begin
3108:C      3   deobvbk := outblock;
3109:C      3   dnumfiles := outmodnum;
3110:C      3   outopen := false; outmodnum := 0;
3111:C      3   lowheap := outdirectory;
3112:C      3   blockwrite(outfile, outdirectory.fdp^, outdirectsize, 0);
3113:C      3   close(outfile, 'lock');
3114:C      3   if ioresult <> 0 then escape(126);
3115:C      3   end;
3116:C      2   end;
3117:S
3118:D      1 procedure openout(boot: boolean);
3119:D      -8 2 var i, j: integer;
3120:D      -18 2 nul: string[i]; typestring: string[6];

```



```

3121:D -30 2   thirdparm: string[10];
3122:C       2   begin
3123:C       2   thirdparm := 'shared';
3124:C       2   if linking then lowheap := infostart;
3125:C       2   linking := false;
3126:C       2   if outopen then
3127:C       3   begin
3128:C       3   close(outfile);
3129:C       3   lowheap := outdirectory;
3130:C       3   end;
3131:C       2   outopen := false;
3132:C       2   fgotoxy(output, 22,4); write(cteol);
3133:C       2   readln(outfile);
3134:C       2   trim(outfile);
3135:C       2   if strlen(outfile) > 0 then
3136:C       3   begin
3137:C       3   nul := '';
3138:C       3   if boot then
3139:C       4   begin
3140:C       4   fixname(outfile, sysfile);
3141:C       4   reset(outfile, outfile); close(outfile, 'PURGE');
3142:C       4   typestring := '.SYSM'; fmakefile(outfile, outfile, thirdparm, typestring);
3143:C       4   outopen := (iresult = 0);
3144:C       4   outblock := 0;
3145:C       4   end
3146:C       4   else
3147:C       4   begin
3148:C       4   fixname(outfile, codefile);
3149:C       4   typestring := '.CODE'; fmakefile(outfile, outfile, thirdparm, typestring);
3150:C       4   if iresult = 0 then
3151:C       5   begin
3152:C       5   gbytes(outdirectory.a, outdirectsize*blocksize);
3153:C       5   with outdirectory.fdp[0] do
3154:C       6   begin
3155:C       6   dfirstblk := 0; dlastblk := outdirectsize;
3156:C       6   dfkind := untypedfile (volume entry);
3157:C       6   moveleft(outfile, dvid, sizeof(volname));
3158:C       6   if strlen(outfile) > vlength then setstrlen(dvid, vlength);
3159:C       6   deovblk := outdirectsize; dnumfiles := 0;
3160:C       6   dloadtime := 0; dlastboot := todaysdate;
3161:C       6   end;
3162:C       5   outblock := outdirectsize;
3163:C       5   outopen := true;
3164:C       5   end;
3165:C       4   end;
3166:C       3   if outopen then booting := boot
3167:C       4   else
3168:C       4   begin
3169:C       4   booting := false;
3170:C       4   errorline;
3171:C       4   write('cannot open ', outfile, ', ', ');
3172:C       4   ioerror;
3173:C       4   end;
3174:C       3   end;
3175:C       2   end; { openout }
3176:S
3177:D -8 1 procedure setmaxmodules;
3178:D       2 var total, excess: integer;
3179:C       2 begin
3180:C       2 fgotoxy(output, 30,6); write(cteol);

```

```

3181:C       2 if readint(maxmodules) then
3182:C       3 begin
3183:C       3 if maxmodules > 300000 then
3184:C       4 begin
3185:C       4 maxmodules := 38;
3186:C       4 escape(125);
3187:C       4 end;
3188:C       3 if maxmodules <= 0 then maxmodules := 0;
3189:C       3 outdirectsize := ((maxmodules*1)*entrysize*(blocksize-1)) div blocksize;
3190:C       3 maxmodules := outdirectsize*blocksize div entrysize - 1;
3191:C       3 end;
3192:C       2 end;
3193:S
3194:D 1 procedure setreloc;
3195:D       2 begin
3196:C       2 fgotoxy(output, 21,7); write(cteol);
3197:C       2 if readint(startreloc) then ;
3198:C       2 end;
3199:S
3200:D 1 procedure setglobal;
3201:D       2 begin
3202:C       2 fgotoxy(output, 21,8); write(cteol);
3203:C       2 if readint(startglobal) then ;
3204:C       2 end;
3205:S
3206:D 1 procedure setcopyright;
3207:D       2 begin
3208:C       2 fgotoxy(output, 0,12); write(cteol);
3209:C       2 fgotoxy(output, 22,11); write(cteol);
3210:C       2 readln(copyright);
3211:C       2 end;
3212:S
3213:D -4 1 procedure makepatchspace;
3214:D       2 var pmod:   addrec;
3215:C       2 begin
3216:C       2 fgotoxy(output, 23,9); write(cteol);
3217:C       2 if readint(patchbytes) then
3218:C       3 if patchbytes > 0 then
3219:C       4 begin
3220:C       4 patchbytes := patchbytes + ord(odd(patchbytes));
3221:C       4 gbytes(pmod.a, sizeof(moduledescriptor, true));
3222:C       4 with pmod.mdp do
3223:C       5 begin
3224:C       5 patchmod := true; patchsize := patchbytes;
3225:C       5 link := newmods; newmods := pmod.mdp;
3226:C       5 patchlink := NIL; patchlist := NIL;
3227:C       5 end;
3228:C       4 end;
3229:C       2 end;
3230:S
3231:D -82 1 procedure setname;
3232:D       2 var s: string80;
3233:C       2 begin
3234:C       2 fgotoxy(output, 24,6); write(cteol);
3235:C       2 readln(s); trim(s);
3236:C       2 if strlen(s)=0 then newmodname.syp := NIL
3237:C       3 else
3238:C       3 begin
3239:C       3 upc(s);
3240:C       3 gbytes(newmodname.a, strlen(s)*2-ord(odd(strlen(s))));

```



```

3241:C      3 moveleft(s, newmodname.syp^, strlen(s)+1);
3242:C      3 end;
3243:C      2 end; {setname}
3244:S
3245:D      1 procedure openmod;
3246:D      2 var s: string80;
3247:D      2 i: shortint;
3248:C      2 begin
3249:C      2 verifying := false; vmodnum := 0;
3250:C      2 fgotoxy(output, 18,18); write(ctypeol);
3251:C      2 readln(s); trim(s);
3252:C      2 if strlen(s) > 0 then
3253:C      2 begin
3254:C      2   upc(s); i := 1;
3255:C      2   while (i <= fdirectory^[0].dnumfiles) and (vmodnum = 0) do
3256:C      2     with fdirectory^[i] do
3257:C      2       begin
3258:C      2         upc(dtid);
3259:C      2         if s = dtid then vmodnum := i
3260:C      2         else i := i + 1;
3261:C      2       end;
3262:C      2     if vmodnum = 0 then
3263:C      2       begin
3264:C      2         errorline; write('module ', s, ' not found in file');
3265:C      2         escape(123);
3266:C      2       end;
3267:C      2     end;
3268:C      2 end; {openmod}
3269:S
3270:D      1 procedure findmod(var s:filename; var n:shortint);
3271:D      2 var i: integer;
3272:C      2 begin
3273:C      2 if strlen(s)=0 then { if no name given then get it }
3274:C      2 begin readln(s); trim(s); end;
3275:C      2 n := 0;
3276:C      2 if strlen(s) > 0 then
3277:C      2 begin
3278:C      2   upc(s); i := 1;
3279:C      2   while (i <= fdirectory^[0].dnumfiles) and (n = 0) do
3280:C      2     with fdirectory^[i] do
3281:C      2       begin
3282:C      2         upc(dtid);
3283:C      2         if s = dtid then n := i else i := i + 1;
3284:C      2       end;
3285:C      2     if n = 0 then n := -1; { signal not found }
3286:C      2   end;
3287:C      2 end; { findmod }
3288:S
3289:D      1 procedure clear(N: shortint);
3290:C      2 begin
3291:C      2 repeat writeln(ctypeol); n := n - 1; until n <= 0;
3292:C      2 end;
3293:S
3294:D      1 procedure none;
3295:C      2 begin write('(none)'); clear(1); end;
3296:S
3297:D      1 procedure doedit;
3298:D      2 var
3299:D      2 firstmodname, untilmodname, oldvmodname: filename;
3300:D      2 firstmodnum, untilmodnum, oldvmodnum: integer;

```

```

3301:D      2 oldfilename: string80;
3302:D      2 modlist: string255;
3303:D      2 assoc: boolean;
3304:D      2 lc: string[4];
3305:D      2 tempf: filename;
3306:D      2 im: shortint;
3307:S
3308:D      2 procedure checkassoc;
3309:C      2 begin
3310:C      2   assoc := assoc and (vmodnum > 0);
3311:C      2   if assoc then
3312:C      2     begin firstmodname := fdirectory^[vmodnum].dtid;
3313:C      2     firstmodnum := vmodnum;
3314:C      2     assoc := assoc and (firstmodnum < untilmodnum);
3315:C      2   end;
3316:C      2 end;
3317:D      2 procedure outoforder;
3318:C      2 begin errorline; write('module ', tempf, ' out of order'); dobeep;
3319:C      2 end;
3320:D      2 procedure mnotfound;
3321:C      2 begin errorline; write('module ', tempf, ' not found'); dobeep;
3322:C      2 end;
3323:S
3324:D      2 begin { doedit }
3325:C      2 untilmodname := '(end of file)';
3326:C      2 untilmodnum := fdirectory^[0].dnumfiles+1;
3327:C      2 if vmodnum=0 then firstmodname := '(none)';
3328:C      2 else firstmodname := fdirectory^[vmodnum].dtid;
3329:C      2 firstmodnum := vmodnum;
3330:C      2 assoc := true;
3331:C      2 fgotoxy(output, 0,2); write(ctypeos);
3332:C      2 repeat
3333:C      2   fgotoxy(output, 0,2);
3334:C      2   writeln('S Stop editing');
3335:C      2   clear(2);
3336:C      2   if firstmodnum > 0 then
3337:C      2     writeln('C Copy First module upto Until module', ctypeol);
3338:C      2   else clear(1);
3339:C      2   writeln('F First module: ', firstmodname, ctypeol);
3340:C      2   writeln('U Until module: ', untilmodname, ctypeol);
3341:C      2   clear(1);
3342:C      2   writeln('A Append module(s)'); clear(5);
3343:C      2   fgotoxy(output, 0,18);
3344:C      2   write('M input Module: ');
3345:C      2   if vmodnum = 0 then begin none; clear(3); end
3346:C      2   else
3347:C      2     begin
3348:C      2       writeln(fdirectory^[vmodnum].dtid, ctypeol);
3349:C      2       if booting then lc := 'boot';
3350:C      2       else if linking then lc := 'link';
3351:C      2       else lc := 'copy';
3352:C      2     end;
3353:C      2   writeln('I Transfer (' lc, ') module', ctypeol);
3354:C      2   writeln('<space> to continue verifying', ctypeol);
3355:C      2 end;
3356:C      2 getcommandchar('Edit option?', commandchar);
3357:C      2 case commandchar of
3358:C      2   'A':begin
3359:C      2     oldfilename := infilename; { save current inputfile name }
3360:C      2     oldvmodnum := vmodnum; { same current module number & name }

```



```
3361:      4      oldvmodname := fdirectory^[vmodnum].dtid;
3362:      4      fgotoxy(output,0,13); write('      Input file: ',cteol);
3363:      4      setstrlen(infile,0);
3364:      4      openin;      { get new input file }
3365:      4      if strlen(infile)>0 then begin      { 3.0 BUG FIX -- 4/11/84 }
3366:      5      { get list of modules and copy them }
3367:      5      fgotoxy(output,0,10);
3368:      5      writeln('enter list of modules or = for all');
3369:      5      readln(modlist); trim(modlist); upc(modlist);
3370:      5      if modlist='=' then
3371:      6      begin { all modules }
3372:      6      for im:=1 to fdirectory^[0].dnumfiles do
3373:      7      begin
3374:      7      fgotoxy(output,0,11); write(fdirectory^[im].dtid,cteol);
3375:      7      copymodule(im);
3376:      7      end;
3377:      6      end
3378:      6      else
3379:      6      while strlen(modlist)>0 do
3380:      7      begin
***WARNING: (line 3381): STRPOS does not conform to HP standard, see $SWITCH_STRPOS
3381:      7      im:=strpos(' ',modlist);
3382:      7      if im=0 then im:=strlen(modlist)+1;
3383:      7      try
3384:      8      if im>sizeof(tempf) then escape(129)
3385:      9      else tempf:=str(modlist,1,im-1);
3386:      8      if im>strlen(modlist) then setstrlen(modlist,0)
3387:      9      else strdelete(modlist,1,im);
3388:      8      if strlen(tempf)>0 then
3389:      8      begin { find the module and copy it }
3390:      9      findmod(tempf,vmodnum);
3391:      8      if vmodnum>0 then copymodule(vmodnum) else escape(123);
3392:      9      fgotoxy(output,0,11); write(modlist,cteol);
3393:      8      end;
3394:      8      recover
3395:      8      begin
3396:      8      im:=escapecode; errorline;
3397:      8      case im of
3398:      9      123: writeln('module ',tempf,' not found');
3399:      9      129: writeln('invalid module name');
3400:      8      otherwise escape(im)
3401:      9      end; { case im }
3402:      8      dobeep; setstrlen(modlist,0);      { zap module list to force exit }
3403:      8      end; { end recover }
3404:      7      end; {while list not empty}
3405:      5
3406:      5      if not streaming then      { 3.0 bug fix -- 4/9/84 jws }
3407:      6      repeat getcommandchar('Append done, <space> to continue',commandchar);
3408:      7      until commandchar=' ';
3409:      5      end;      { 3.0 BUG FIX -- 4/11/84 }
3410:      5
3411:      4      infile:= oldfilename;
3412:      4      openin;      { reopen the old file & find old input module }
3413:      4      if oldvmodnum=0 then vmodnum:=0
3414:      5      else
3415:      5      begin
3416:      5      findmod(oldvmodname,vmodnum);
3417:      5      if (vmodnum<>oldvmodnum) then
3418:      6      begin errorline;
3419:      6      write('unable to find old input module ',oldvmodname);
```

```
3420:      6      vmodnum:=0; dobeep;
3421:      5      end;
3422:      5      end;
3423:      4      end;
3424:      4      'C':if (firstmodnum>0) and (firstmodnum<untilmodnum) then
3425:      5      begin
3426:      5      for im:=firstmodnum to untilmodnum-1 do
3427:      6      begin fgotoxy(output,0,8);
3428:      6      write('now copying ',fdirectory^[im].dtid,cteol);
3429:      6      copymodule(im);
3430:      6      end;
3431:      5      if assoc then
3432:      6      begin
3433:      6      if untilmodnum>fdirectory^[0].dnumfiles
3434:      7      then begin vmodnum := 0;
3435:      7      firstmodname := '(none)';
3436:      7      end
3437:      7      else begin vmodnum := untilmodnum;
3438:      7      firstmodname := fdirectory^[vmodnum].dtid;
3439:      7      end;
3440:      6      firstmodnum := vmodnum;
3441:      6      assoc:= assoc and (vmodnum>0);
3442:      6      end;
3443:      5      end
3444:      5      else dobeep;
3445:      4      'F':begin
3446:      4      fgotoxy(output,17,6); write(cteol);
3447:      4      setstrlen(tempf,0); findmod(tempf,im);
3448:      4      case im of
3449:      5      -1: notfound;
3450:      5      0: { no module name given, so use default }
3451:      5      begin if (vmodnum>0) and (vmodnum<untilmodnum) then
3452:      6      begin firstmodname := fdirectory^[vmodnum].dtid;
3453:      6      firstmodnum := vmodnum;
3454:      6      assoc := assoc and (vmodnum<untilmodnum);
3455:      6      end
3456:      6      else dobeep;
3457:      5      end;
3458:      5      otherwise { found the module }
3459:      5      if im<untilmodnum then
3460:      6      begin firstmodname := tempf; firstmodnum:=im; assoc:=false;
3461:      6      end
3462:      6      else outoforder;
3463:      5      end; { case im }
3464:      4      end;
3465:      4      'M': if fdirectory<>NIL then
3466:      5      begin openmod;
3467:      5      if vmodnum>0 then begin      { 3.0 BUG # 57 4/10/84 }
3468:      6      assoc := assoc and (vmodnum<untilmodnum);
3469:      6      if assoc then
3470:      7      begin
3471:      7      firstmodname := fdirectory^[vmodnum].dtid;
3472:      7      firstmodnum := vmodnum;
3473:      7      end;
3474:      6      end
3475:      6      else dobeep      { 3.0 BUG # 57 4/10/84 }
3476:      6      end
3477:      5      else dobeep;      { 3.0 BUG # 57 4/10/84 }
3478:      4      'S';
3479:      4      'T': if vmodnum>0 then
```



```

3480:C      5      begin xfer; checkassoc;
3481:C      5      end
3482:C      4      else dobeep;
3483:C      4      'U':begin
3484:C      4          fgotoxy(output,17,7); write(ctypeol);
3485:C      4          setstriein(tempf,0); findmod(tempf,im);
3486:C      4          case im of
3487:C      5              -1: mnotfound;
3488:C      5              0: { no module name given, so default };
3489:C      5              begin untilmodname := '(end of file)';
3490:C      5                  untilmodnum := fdirectory^0].dnumfiles+1;
3491:C      5              end;
3492:C      5              otherwise { found the module }
3493:C      5                  if im>firstmodnum then
3494:C      6                      begin untilmodname := tempf; untilmodnum:=im;
3495:C      6                      end
3496:C      6                  else outoforder;
3497:C      6                  assoc := assoc and (im>firstmodnum);
3498:C      5              end; { case im }
3499:C      4      end;
3500:C      4      ' ': if vmodnum>0 then
3501:C      5          begin verifynext; checkassoc;
3502:C      5          end
3503:C      5          else dobeep;
3504:C      4          otherwise dobeep;
3505:C      4      end;
3506:C      2      until commandchar='S';
3507:C      2      end; { doedit }
3508:S
3509:D      1 procedure finishboot;
3510:C      2      begin
3511:C      2          close(outfile, 'LDCK');
3512:C      2          if ioresult<>0 then escape(126);
3513:C      2          outopen := false; booting := false;
3514:C      2          outmodnum := 0;
3515:C      2          end;
3516:S
3517:D      1 procedure menu;
3518:D      -6      2 var lc: string[4];
3519:C      2      begin
3520:C      2          fgotoxy(output, 0,2); write('Q Quit'); clear(1);
3521:C      2          write('P Printout ');
3522:C      2          if printopen then
3523:C      3              begin
3524:C      3                  if printeron then write('ON ');
3525:C      3                  else write('OFF ');
3526:C      3                  write(listfilename);
3527:C      3                  clear(1);
3528:C      3                  end
3529:C      3              else none;
3530:C      2          end;
3531:C      2          if outmodnum > 0 then write('K Keep o')
3532:C      2          else if (newmods = NIL) and not booting
3533:C      3              then write('O ');
3534:C      3              else write(' ');
3535:C      2          write('Output file: ');
3536:C      2          if outopen then writeln(outfile,ctypeol)
3537:C      3              else none;
3538:C      2          if outopen then
3539:C      2

```

```

3540:C      3      begin
3541:C      3          if booting then write('B finish Boot')
3542:C      3          else if newmods <> NIL then write('L finish Linking')
3543:C      3          else if linking then write('C Copy')
3544:C      3          else write('L Link');
3545:C      3          write(ctypeol); fgotoxy(output, 40,5);
3546:C      3          if booting then
3547:C      4              begin
3548:C      4                  writeln('8DOTING');
3549:C      4                  lc := 'boot';
3550:C      4                  end
3551:C      4              else if linking then begin
3552:C      4                  writeln('LINKING');
3553:C      4                  lc := 'link';
3554:C      4                  end
3555:C      4              else
3556:C      4                  begin
3557:C      4                      writeln('CDPYING');
3558:C      4                      lc := 'copy';
3559:C      4                  end
3560:C      3          end
3561:C      3          begin
3562:C      3              writeln('B write to Boot disk',ctypeol);
3563:C      3              writeln('H file Header maximum size: ',maxmodules,ctypeol);
3564:C      3          end;
3565:S
3566:C      2      if linking and outopen then
3567:C      3          begin
3568:C      3              write('N Name of new module: ');
3569:C      3              if newmodname.syp = NIL then none
3570:C      3              else writeln(newmodname.syp,ctypeol);
3571:C      3              writeln('R Relocation base: ',startreloc:12,ctypeol);
3572:C      3              writeln('G Global base: ',startglobal:12,ctypeol);
3573:C      3              write('S Space for patches:');
3574:C      3              if patchbytes > 0 then writeln(patchbytes:12)
3575:C      3              else clear(1);
3576:C      3              patchbytes := 0;
3577:C      3              write('D output Def table? ');
3578:C      3              if defsout then write('YES')
3579:C      3              else write('NO');
3580:C      3              writeln(ctypeol);
3581:C      3              writeln('X copyright notice: ',copyright);
3582:C      3          end
3583:C      3          else clear(7);
3584:C      2      fgotoxy(output, 0,13);
3585:C      2      write('I Input file: ');
3586:C      2      if fdirectory = NIL then
3587:C      3          begin none; clear(7); end
3588:C      3          else
3589:C      3              begin
3590:C      3                  writeln(infile,ctypeol);
3591:C      3                  if outopen then writeln('E Edit') else clear(1);
3592:C      3                  writeln('F list File directory');
3593:C      3                  if outopen then writeln('A ',lc, ' All modules') else clear(1);
3594:C      3                  write('V Verify modules');
3595:C      3                  if verifying then
3596:C      4                      begin
3597:C      4                          fgotoxy(output, 40,17); writeln('VERIFYING');
3598:C      4                      end
3599:C      3                  else clear(1);

```



```

3600:C      3 write('M input Module: ');
3601:C      3 if vmodnum = 0 then
3602:C      4 begin none; clear(3); end
3603:C      4 else
3604:C      4 begin
3605:C      4 writein(fdirectory^vmodnum].drid,cteol);
3606:C      4 writein('U Unassemble object');
3607:C      4 if outopen then writein('f Transfer ('lc,') module')
3608:C      5 else clear(1);
3609:C      4 if verifying then writein('<space> to continue verifying')
3610:C      5 else clear(1);
3611:C      4 end;
3612:C      3 end;
3613:C      2 end; {menu}
3614:C
3615:C      1 procedure getcommand;
3616:C      -82 2 var err: string[80];
3617:C      2 begin
3618:C      2 repeat
3619:C      3 try
3620:C      4 menu;
3621:C      4 getcommandchar('command?',commandchar);
3622:C      4 case commandchar of
3623:C      5 'A': if (fdirectory<>NIL) and outopen then copyfile else dobeep;
3624:C      5 'B': if booting then finishboot
3625:C      6 else if outopen then dobeep
3626:C      7 else openout(true);
3627:C      5 'C': if outopen and linking and (newmods = NIL)
3628:C      6 and not booting then copyon else dobeep;
3629:C      5 'D': if linking and outopen
3630:C      6 then defsout := not defsout else dobeep;
3631:C      5 'E': if (fdirectory<>NIL) and outopen then doedit else dobeep;
3632:C      5 'F': if fdirectory<>NIL then printdirectory else dobeep;
3633:C      5 'G': if linking and outopen then setglobal else dobeep;
3634:C      5 'H': if outopen then dobeep else setmaxmodules;
3635:C      5 'I': begin setstrlen(infile,0); openin; end;
3636:C      5 'K': if (outmodnum > 0) and not booting
3637:C      6 then closeout else dobeep;
3638:C      5 'L': if booting then dobeep
3639:C      6 else if newmods <> NIL then link
3640:C      7 else if outopen and not linking
3641:C      8 then initlink else dobeep;
3642:C      5 'M': if fdirectory<>NIL then openmod else dobeep;
3643:C      5 'N': if linking and outopen then setname else dobeep;
3644:C      5 'O': if (outmodnum = 0) and (newmods=NIL) and not booting
3645:C      6 then openout(false) else dobeep;
3646:C      5 'P': toggleprinter;
3647:C      5 'Q': quit;
3648:C      5 'R': if linking and outopen then setreloc else dobeep;
3649:C      5 'S': if linking and outopen then makepatchspace else dobeep;
3650:C      5 'T': if (vmodnum = 0) and outopen then xfer else dobeep;
3651:C      5 'U': if (vmodnum > 0) then unassemble else dobeep;
3652:C      5 'V': if fdirectory<>NIL then verifmod else dobeep;
3653:C      5 'X': if linking and outopen then setcopyright else dobeep;
3654:C      5 ' ': if verifying then verifnext;
3655:C      5 otherwise dobeep;
3656:C      5 end;
3657:C      2 end;
3658:C
3659:C      4 ecover

```

```

3660:C      4 begin
3661:C      4 if (escapecode <> -20) and (escapecode <> 123) and (escapecode<>128)
3662:C      5 then errorline;
3663:C      4 if escapecode=-10 then begin getioerrmsg(err, ires); writein(err); end
3664:C      5 else case escapecode of
3665:C      6 110: write('symbols defined recursively');
3666:C      6 111: write('improper link info format');
3667:C      6 112: write('not enough memory');
3668:C      6 113: write('output file full');
3669:C      6 114: write('error writing to boot disk, ioresult = 'ires:1);
3670:C      6 116: write(' ', infile, ' is not a code file');
3671:C      6 118: write('printer or list file not on line');
3672:C      6 119: write('duplicate symbol definition');
3673:C      6 120: write('module being booted has external references');
3674:C      6 121: write('unexpected end of code');
3675:C      6 122: write('errors during linking',cteol);
3676:C      6 123,128,129: (error message already printed);
3677:C      6 124: write('integer required');
3678:C      6 125: write('integer too large');
3679:C      6 126: write('unable to close output, ioresult = 'ires:1);
3680:C      6 127: write('file header full');
3681:C      6 otherwise escape(escapecode);
3682:C      6 end; (case escapecode)
3683:C      4 if streaming then escape(-1);
3684:C      4 if (escapecode-100) in [12,16] then closein;
3685:C      4 if (escapecode-100) in [10,13,19,22,26,28] then
3686:C      5 begin
3687:C      5 if newmods <> NIL then begin closein; closefiles; end;
3688:C      5 linking := false; newmods := NIL;
3689:C      5 if outopen then close(outfile);
3690:C      5 outopen := false; outmodnum := 0;
3691:C      5 booting := false;
3692:C      5 lowheap := lowheap0;
3693:C      5 end;
3694:C      4 end; {recover}
3695:C      3 until commandchar = 'Q';
3696:C      2 end {getcommand};
3697:C
3698:C
3699:C      1 procedure wrapup;
3700:C      2 begin
3701:C      2 pageeject;
3702:C      2 closein;
3703:C      2 closefiles;
3704:C      2 if (pagenum=0) and (linenum=0)
3705:C      3 then close(listing)
3706:C      3 else close(listing, 'lock');
3707:C      2 end; {wrapup}
3708:C
3709:C      1 begin {program linker}
3710:C      1 with linkerdate do
3711:C      2 begin day := 4; year := 84; month := 6; end;
3712:C      1 sysdate(todaydate);
3713:C
3714:C      1 pagenum := 0; linenum := 0;
3715:C
3716:C      1 fgotoxy(output, 0,0);
3717:C      1 printheader(output);
3718:C      1 fgotoxy(output, 0, 22);
3719:C      1 writein('Copyright 1984 Hewlett-Packard Company.');
```



```
3720:S
3721:C      1  mark(lowheap.p); lowheap0 := lowheap;
3722:C      1  highheap.a := lowheap.a + memavail - 5000;
3723:C      1  release(highheap.p); highheap0 := highheap;
3724:S
3725:C      1  listfilename := 'PRINTER:LINK.ASC';
3726:C      1  rewrite(listing, listfilename);
3727:C      1  printopen := ioreult = 0;
3728:C      1  printeron := false;
3729:S
3730:C      1  patchbytes := 0;
3731:C      1  maxmodules := 38;      outdirectsize := 2;
3732:C      1  linking := false;    newmods := NIL;
3733:C      1  outopen := false;    outmodnum := 0;
3734:C      1  booting := false;    verifying := false;
3735:C      1  fdirectory := NIL;   vmodnum := 0;
3736:C      1  loadfib.php := NIL;
3737:S
3738:C      1  try getcommand;
3739:S
3740:C      2  recover begin
3741:C      2      esccode := escapecode;
3742:C      2      wrapup;
3743:C      2      escape(esccode);
3744:C      2      end;
3745:S
3746:C      1  wrapup;
3747:C
3748:C      1  end.
```

No errors. 1 warnings.

***** Nonstandard language features enabled *****

LIFDAM

Description

LIFDAM provides access to LIF format directories. The module also contains the code which installs itself in the system.

Usage

LIFDAM is called by the filesystem to operate on LIF format directories.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1982.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S     RESTRICTED RIGHTS LEGEND
11:S
12:S     Use, duplication, or disclosure by the Government
13:S     is subject to restrictions as set forth in
14:S     paragraph (b) (3) (B) of the Rights in Technical
15:S     Data and Computer Software clause in
16:S     DAR 7-104.9(a).
17:S
18:S     HEWLETT-PACKARD COMPANY
19:D     0 Fort Collins, Colorado      *)
20:S
21:S     0 $MODCLS
22:D     0 $DERUG OFF, range off, ovflcheck off$
23:D     0 program instliffdam;
24:D     1 module lifmodule;
25:D     1 import sysglobals, sysdevs, misc, fs;
26:D     1 export
27:D     1 procedure lifdam(anyvar f: fib; unum: unitnum; request: damrequesttype);
28:D     1 procedure installiffdam;
29:D     1
30:S
31:D     1 implement
32:D     1 const
33:D     1   entrysize = 32;
34:D     1   type
35:D     1     vname = packed array[1..6] of char;
36:D     1     lifname = packed array[1..10] of char;
37:D     1     bcd = 0..15;
38:D     1     word = 0..65535;
39:D     1     integer16 = 32768..32767;
40:D     1     word15 = 0..32767;
41:D     1     tdate = packed array[1..12] of bcd;
42:D     1     lvheader = packed record (volume header sector 0)
43:D     1       discid : word;
44:D     1       volname : vname;
45:D     1       dstart : integer;
46:D     1       dummy1 : integer16;
47:D     1       dummy2 : integer16;
48:D     1       dsize : integer;
49:D     1       version : integer16;
50:D     1       dummy3 : integer16;
51:D     1       tps : integer; (tracks/surface)
52:D     1       spm : integer; (surfaces/medium)
53:D     1       spt : integer; (sectors/track)
54:D     1       cdate : tdate; (volume create time)
55:D     1       filler : packed array[21..123] of integer16;
56:D     1       sdate : tdate;
57:D     1       dummy4 : integer16;
58:D     1     end;
59:D     1   direntry = packed record
60:D     1     fname : lifname;

```

```

61:D     1   ftype : integer16;
62:D     1   fstart : integer;
63:D     1   fsize : integer;
64:D     1   fdate : tdate;
65:D     1   lastvol : boolean;
66:D     1   volnumber: word15;
67:D     1   extension: integer;
68:D     1   end;
69:S
70:D     1   spacerec = record
71:D     1     sstart : integer;
72:D     1     ssize : integer;
73:D     1     here : integer;
74:D     1     hole : integer;
75:D     1   end;
76:D     1   catarray = array[1..maxint] of catentry;
77:D     1   dirfile = file of direntry;
78:D     1   var
79:D     -4 1   dir : ^dirfile;
80:S
81:D     -4 1 (*****
82:D     1   procedure goodio;
83:C     2   begin if ioresult<>ord(inoerror) then escape(0); end;
84:S
85:D     -4 1 (*****
86:D     1   procedure badio(result : iorsltwd);
87:C     2   begin ioresult := ord(result); escape(0); end;
88:S
89:D     -4 1 (*****
90:D     1   procedure pactostr(anyvar pc: lifname; l:integer; var s:string);
91:D     -4 2   var i : integer;
92:C     2   begin
93:C     2     setstrlen(s,1); for i:=1 to l do s[i] := pc[i]; i := 1;
94:C     2     while (i>1) and (s[i]=' ') do i:=i-1; setstrlen(s,i);
95:C     2   end;
96:S
97:D     -4 1 (*****
98:D     1   procedure strtovac(anyvar s:string255; l:integer;
99:D     2   var i,k : integer; anyvar pc:lifname; sizechk:boolean);
100:D     -8 2   begin
101:C     2     if sizechk then
102:C     3     if (strlen(s)>1) or (strlen(s)=0) then badio(ibadtitle);
103:C     2     k:=strlen(s);
104:C     2     for i:=1 to l do if i>k then pc[i] := ' ' else pc[i] := s[i];
105:C     2   end;
106:S
107:D     -4 1 (*****
108:D     1   procedure strttoany(var s:string; anyvar s2:string255);
109:D     2   begin s2:=s; end;
110:C
111:S
112:D     -4 1 (*****
113:D     1   procedure setdate(var d:tdate);
114:D     2   var
115:D     -6 2   doy:daterec; tod:t1mrec;
116:C     2   begin
117:C     2     sysdate(doy); systime(tod);
118:C     2     with doy, tod do
119:C     3     begin
120:C     3       d[1] := year div 10; d[2] := year mod 10;

```



```

121:C      3      d[3] := month div 10; d[4] := month mod 10;
122:C      3      d[5] := day div 10; d[6] := day mod 10;
123:C      3      d[7] := hour div 10; d[8] := hour mod 10;
124:C      3      d[9] := minute div 10; d[10] := minute mod 10;
125:C      3      d[11] := (centisecond div 100) div 10;
126:C      3      d[12] := (centisecond div 100) mod 10;
127:C      3      end;
128:C      2      end;
129:S
130:D      -4 1 {*****}
131:D      1      procedure lifnametostr(anyvar ln:lifname; var s:string);
132:D      2      label 1;
133:D      2      var
134:D      -4 2      sl : integer;
135:D      -6 2      fk : filekind;
136:D      -7 2      found : boolean;
137:S
138:C      2      begin (lifname to str)
139:C      2      pactostr(ln,10,s); sl := strlen(s);
140:C      2      if sl=10 then
141:C      3      begin
142:C      3      if suffix(s)=datafile then
143:C      4      begin { rip underscores and try to add suffix }
144:C      4      while (sl>=1) and (s[sl]='_') do sl := sl - 1;
145:C      4      for fk:=untypedfile to lastfilekind do
146:C      5      begin
147:C      5      if strlen(suffixtable^[fk])>0 then
148:C      6      if suffixtable^[fk][1]=s[sl] then
149:C      7      begin { round suffix }
150:C      7      { change last char to . then append suffix }
151:C      7      setstrlen(s,sl); s[sl] := '.';
152:C      7      s := s + suffixtable^[fk]; goto 1;
153:C      7      end;
154:C      5      end; { for }
155:C      4      end;
156:C      3      end;
157:C      2      1:end; {lifname to str}
158:S
159:D      -4 1 {*****}
160:D      1      procedure strtolifname(var s:string; var ln:lifname);
161:D      2      var
162:D      -8 2      sl, i : integer;
163:D      -252 2      temp,temp2 : ftd; {31jan83 temp2 for case insensitive suffix}
164:D      -254 2      fk : filekind;
165:S
166:C      2      begin {str to lifname}
167:C      2      sl := strlen(s);
168:C      2      fk := suffix(s);
169:C      2      if fk=datafile then
170:C      3      begin { data files have no suffix }
171:C      3      if sl>10 then badio(ibadtitle);
172:C      3      strtovac(s,10,ln,false);
173:C      3      end
174:C      3      else
175:C      3      begin { remove the suffix }
176:C      3      sl := strlen(s)-strlen(suffixtable^[fk]);
177:C      3      if sl>10 then badio(ibadtitle);
178:C      3      strtovac(s,10,ln,false); { pack the name }
179:C      3      { replace dot with first char of suffix (to preserve uniqueness)}
180:C      3      ln[sl] := suffixtable^[fk][1]; sl := sl+1;

```

```

181:C      3      for i:=sl to 10 do ln[i] := '_'; { pad with _ }
182:C      3      end;
183:C      2      lifnametostr(ln,temp); { decompress the name as a final check }
184:C      2      if stemp<>s then
185:C      3      begin {31jan83 case insensitive suffix testing}
186:C      3      temp2 := s; { copy s, then remove given suffix }
187:C      3      setstrlen(temp2,strlen(temp2)-strlen(suffixtable^[fk]));
188:C      3      temp2 := temp2 + suffixtable^[fk]; { add suffix from table }
189:C      3      if stemp<>temp2 then badio(ibadtitle); { check again }
190:C      3      end;
191:C      2      end; {str to lifname}
192:S
193:D      -4 1 {*****}
194:D      -4 1 {*****}
195:D      1      procedure lifdam(anyvar f: fib; unum: unitnum; request: damrequesttype);
196:D      2      var
197:D      -256 2      vol : lvheader;
198:D      -274 2      valid : vid;
199:D      -277 2      ok, mediavalid, anychange: boolean;
200:D      -294 2      dindex, dlast, dend, vsize : integer;
201:D      -326 2      dentry : direntry;
202:S
203:D      -326 2      $iocheck off$
204:D      -326 2      {*****}
205:D      2      function volsize:integer;
206:C      3      begin
207:C      3      if vsize=0 then vsize := ueovbytes(unum);
208:C      3      vsize := vsize;
209:C      3      end;
210:S
211:D      -326 2      {*****}
212:D      2      procedure cleanup;
213:C      3      begin
214:C      3      if ioresult=ord(zmediumchanged) then mediavalid := false;
215:C      3      unitable^[unum].umediavalid := mediavalid;
216:C      3      unitable^[unum].ureportchange := true;
217:C      3      end;
218:S
219:D      -326 2      {*****}
220:D      2      procedure checkftitle;
221:C      3      begin
222:C      3      if (strlen(f.ftitle)>tidlength) or (strlen(f.ftitle)=0) then badio(ibadtitle);
223:C      3      f.ftid := f.ftitle;
224:C      3      end;
225:S
226:D      -326 2      {*****}
227:D      2      function vvname:boolean;
228:D      3      var
229:D      -4 3      i : integer;
230:D      -5 3      b : boolean;
231:C      3      begin
232:C      3      vvname := true; b := true;
233:C      3      for i := 1 to 6 do {ifeb83 allow all blank names}
234:C      4      begin
235:C      4      if b then b := vol.volname[i]>' ';
236:C      4      if not b then {ifeb83 allow all blank names}
237:C      5      if vol.volname[i]<>' ' then vvname := false;
238:C      4      end
239:C      4      end;
240:S

```



```

241:D -326 2 (*****
242:D 2 function lifvol:boolean;
243:D 3 var
244:D -4 3 i : integer; (31jan83 allow all blank volname)
245:C 3 begin ( read and validate the volume header )
246:C 3 with fibp(dir)^, unitable^[unum] do
247:C 4 begin
248:C 4 fileid := 0; fpeof := maxint; ( initialize dir )
249:C 4 if uisbikd then
250:C 5 begin
251:C 5 call(tm,fibp(dir),readbytes,vol,sizeof(lvheader),0);
252:C 5 with vol do
253:C 5 ok := ((ioresult=ord(inoerror)) and (discid=32768) and
254:C 5 (dummy1=4096) and (dummy2=0) and (dummy3=0) and
255:C 5 (dstart>1) and (dsize>0) and (vname));
256:C 5 end
257:C 5 else ok := false;
258:C 4 ureportchange := true; ( now let TM report any mediachanges )
259:C 4 umediavalid := true;
260:C 4 lifvol := ok;
261:C 4 if ok then
262:C 5 begin
263:C 5 if vol.volname[1]=' ' then (31jan83 allow all blank volname)
264:C 6 begin
265:C 6 setstrlen(volid,6); for i:=1 to 6 do volid[i]:=' ';
266:C 6 end
267:C 5 else
268:C 5 pactostr(vol.volname,6,volid);
269:C 5 if volid<>uvid then
270:C 6 begin mediavalid := false; uvid := volid; end;
271:C 5 end
272:C 5 else setstrlen(uvid,0);
273:C 4 if (not ok) and (ioresult=ord(inoerror)) then ioresult:=ord(inodirectory);
274:C 4 end;
275:C 3 end; ( lifvol )
276:S
277:D -326 2 $iocheck on$
278:D -326 2 (*****
279:D 2 procedure opendir;
280:C 3 begin
281:C 3 if lifvol then
282:C 4 begin
283:C 4 dlast:=vol.dsize * 8; ( # entries in directory )
284:C 4 dendi:=vol.dstart + vol.dsize; ( directory end sector + 1 )
285:C 4 with fibp(dir)^ do
286:C 5 begin ( initialize the fib ( fake OPEN ))
287:C 5 fisNew:=false;
288:C 5 freadable:=true; fwriteable:=true;
289:C 5 freadmode:=false; fbufvalid:=false;
290:C 5 feof:=false; fmodified:=false;
291:C 5 fileid:=vol.dstart*256;
292:C 5 if vol.version>0 then vsize := vol.tps*vol.spm*vol.spt*256
293:C 5 else vsize := 0;
294:C 5 fpeof:=vol.dsize*256; ( end of directory )
295:C 5 fleof:=fpeof;
296:C 5 fpos:=0; am:=amtable^[datafile];
297:C 5 freptcnt:=0; flastpos:=1;
298:C 5 fbufchanged:=false; fbuffered:=true;
299:C 5 end;
300:C 4 dindex:=1; read(dir^,dentry);

```

```

301:C 4 goodio;
302:C 4 end
303:C 4 else escape(0);
304:C 3 end; ( opendir )
305:S
306:D -326 2 (*****
307:D 2 procedure checkvolid;
308:C 3 begin
309:C 3 if f.fvid<>volid then badio(ilostunit);
310:C 3 end;
311:S
312:D -326 2 (*****
313:D 2 procedure flushdir;
314:C 3 begin ( June 83 fixed to account for ops which don't open the directory RDQ )
315:C 3 if fibp(dir)^.freadable
316:C 3 ( directory open so flush thru AM )
317:C 4 then call(fibp(dir)^.am,fibp(dir),flush,ioresult,0,0)
318:C 4 ( directory not open so flush thru TM )
319:C 4 else call(unitable^[unum].tm,fibp(dir),flush,ioresult,0,0);
320:C 3 anychange := false;
321:C 3 end;
322:S
323:D -326 2 (*****
324:D 2 procedure getsdate(anyvar svdate:daterec);
325:C 3 begin
326:C 3 if lifvol then
327:C 4 with vol, svdate do
328:C 5 begin
329:C 5 if not ((sdate[5]=0) and (sdate[8]=0)) then
330:C 6 begin
331:C 6 year:=sdate[1]*10+sdate[2];
332:C 6 month:=sdate[3]*10+sdate[4];
333:C 6 day:=sdate[5]*10+sdate[6];
334:C 6 end;
335:C 5 end;
336:C 3 end;
337:S
338:D -326 2 (*****
339:D 2 procedure setsdate(anyvar svdate:daterec);
340:D 3 var
341:D -4 3 i : integer;
342:C 3 begin
343:C 3 if lifvol then
344:C 4 with svdate, vol do
345:C 5 begin
346:C 5 sdate[1]:=year div 10; sdate[2]:=year mod 10;
347:C 5 sdate[3]:=month div 10; sdate[4]:=month mod 10;
348:C 5 sdate[5]:=day div 10; sdate[6]:=day mod 10;
349:C 5 for i:=7 to 12 do sdate[i]:=0; (clear time of day)
350:C 5 with unitable^[unum] do
351:C 6 begin
352:C 6 call(tm,fibp(dir),writebytes,vol,sizeof(lvheader),0);
353:C 6 goodio; anychange := true;
354:C 6 end;
355:C 5 end;
356:C 3 end; ( setsdate )
357:S
358:D -326 2 (*****
359:D 2 procedure cleandir;
360:D 3 var

```



```

361:D -4 3 k : integer;
362:D -36 3 tempd : direntry;
363:S
364:C 3 begin {cleandir}
365:C 3 k:=1; seek(dir^,k);
366:C 3 repeat
367:C 4 read(dir^,tempd);
368:C 4 if tempd.ftype=-1 then k:=dlast
369:C 5 else
370:C 6 if tempd.ftype<>0 then
371:C 7 begin
372:C 8 if (tempd.fdate[3]=9) and (tempd.fdate[4]=9) then
373:C 9 begin { temp file so purge it }
374:C 10 tempd.ftype:=0;
375:C 11 writedir(dir^,k,tempd); anychange := true;
376:C 12 end;
377:C 13 k:=k+1;
378:C 14 until k>dlast;
379:C 15 mediavalid := true;
380:C 16 end; {clean dir}
381:C 3
382:S
383:D -326 2 (*****
384:D 2 procedure crunchv; { assumed to be called from procedure lifdam only 24jan83}
385:D 3
386:D -12 3 var
387:D -24 3 frompos, topos, todindex : integer;
388:D -32 3 bsize, filesize, movesize : integer;
389:D -33 3 bufptr, heapmark : windowp;
390:D -38 3 changed : boolean;
391:D 3 datafib : fibp;
392:C 3
393:C 3 begin
394:C 3 opendir; checkvolid;
395:C 3 if strlen(f.ftitle)<>0 then badio(ibadtitle);
396:C 3 cleandir;
397:C 3 { allocate the buffer space }
398:C 3 MARK(heapmark);
399:C 3 try
400:C 4 bsize:=(memavail-(1024*5));
401:C 5 if bsize>=256 then
402:C 6 begin { buffer in sector multiples }
403:C 7 bsize:=(bsize div 256) * 256;
404:C 8 newwords(bufptr,bsize div 2);
405:C 9 new(datafib); { set up the data fib }
406:C 10 with datafib^ do
407:C 11 begin
408:C 12 funit:=fibp(dir)^.funit;
409:C 13 fileid:=0;
410:C 14 fpeof:=volsize; fleof:=fpeof;
411:C 15 end;
412:C 16 else escape(-2); { not enough room to run }
413:C 17 { krunch it }
414:C 18 dindex:=1;
415:C 19 todindex:=0; topos:=dend*256; anychange:=false;
416:C 20 repeat
417:C 21 changed:=false;
418:C 22 readdir(dir^,dindex,dentry);
419:C 23 if dentry.ftype=-1 then dindex:=dlast
420:C 24 else

```

```

421:C 6 if dentry.ftype=efitable^[badfile] then
422:C 7 begin { bad sectors ; don't move this file }
423:C 8 todindex:=todindex+1;
424:C 9 if dindex<>todindex then changed:=true; { move the entry }
425:C 10 topos:=(dentry.fstart+dentry.size)*256; { move topos }
426:C 11 end
427:C 12 else
428:C 13 if dentry.ftype=0 then anychange := true { found purged entry }
429:C 14 else
430:C 15 begin { move the file ? }
431:C 16 todindex:=todindex+1;
432:C 17 if dindex<>todindex then changed:=true; { move the entry }
433:S
434:C 18 frompos:=dentry.fstart*256;
435:C 19 if frompos<>topos then
436:C 20 begin { move the data }
437:C 21 filesize:=dentry.fsize*256; { bytes to move }
438:C 22 dentry.fstart:=topos div 256; { set start }
439:C 23 changed:=true;
440:C 24 with unitable^[datafib^.funit] do
441:C 25 repeat
442:C 26 if filesize>bsize then movesize:=bsize
443:C 27 else movesize:=filesize;
444:C 28 call(tm,datafib,readbytes,bufptr^,movesize,frompos);
445:C 29 frompos:=frompos+movesize; goodio;
446:S
447:C 30 call(tm,datafib,writebytes,bufptr^,movesize,topos);
448:C 31 topos:=topos+movesize; goodio;
449:S
450:C 32 filesize:=filesize-movesize;
451:C 33 until filesize=0;
452:C 34 end { move the data }
453:C 35 else topos:=topos+dentry.fsize*256;
454:C 36 end; { move the file ? }
455:C 37 if changed then
456:C 38 begin
457:C 39 writedir(dir^,todindex,dentry); anychange := true;
458:C 40 end;
459:C 41 dindex:=dindex+1;
460:C 42 until dindex>dlast;
461:C 43 if anychange then
462:C 44 begin { put end of directory mark }
463:C 45 if todindex<dlast then
464:C 46 begin
465:C 47 dentry.ftype:=-1;
466:C 48 writedir(dir^,todindex+1,dentry);
467:C 49 end;
468:C 50 end;
469:C 51 call(unitable^[datafib^.funit].tm,datafib,flush,ioreult,0,0);
470:C 52 RELEASE(heapmark);
471:C 53 recover
472:C 54 begin
473:C 55 RELEASE(heapmark);
474:C 56 frompos := ioreult; topos := escapecode; { save state 24jan83 }
475:C 57 if anychange then flushdir; { try to clean up 24jan83 }
476:C 58 ioreult := frompos; { restore the state 24jan83 }
477:C 59 escape(frompos); { exit 24jan83 }
478:C 60 end;
479:C 61 end; {crunchv}
480:S

```



```
481:D -326 2 (*****  
482:D 2 procedure domakedirectory(anyvar cat:catentry);  
483:D 2 var  
484:D -4 3 i : integer;  
485:D -8 3 actualsize : integer;  
486:D -264 3 secbuf : packed array[0..63] of integer;  
487:C 3 begin  
488:C 3 if strlen(f.ftitle)>0 then badio(ibadrequest);  
489:C 3 with vol, cat do  
490:C 4 begin  
491:C 4 if not lifvol then  
492:C 5 begin  
493:C 5 if (ioresult<>ord(inoerror)) and  
494:C 6 (ioresult<>ord(inodirectory)) then escape(0);  
495:C 5 ioresult := ord(inoerror);  
496:C 5 { clear header fields }  
497:C 5 discid := 32768;  
498:C 5 dummy1 := 4096; dummy2 := 0; dummy3 := 0;  
499:C 5 version := 0;  
500:C 5 end  
501:C 5 else checkvalid;  
502:C 4 { directory size checks }  
503:C 4 dstart := 2;  
504:C 4 dsize := ((cextral*entriysize)+255) div 256;  
505:C 4 if dsize<=0 then dsize := 10; { default directory size }  
506:C 4 { size checks }  
507:C 4 actualsize := ueovbytes(unum);  
508:C 4 if (cpsize>actualsize) or (cpsize<1024) then badio(inoroom);  
509:C 4 if (dstart+dsize+1)*256>cpsize then badio(inoroom);  
510:C 4 { fill in the pieces }  
511:C 4 strtovac(cname,6,volname,true); { volume name }  
512:C 4 if version>0 then  
513:C 5 if (tps*spm*spt*256)<>cpsize then version := 0;  
514:S 4  
515:C 4 setdate(cdate); { fill in create date }  
516:C 4 for i := 1 to 12 do sdate[i] := 0; { clear system date }  
517:S 4  
518:C 4 if version=0 then  
519:C 5 { create pseudo level 1 header }  
520:C 5 begin  
521:C 5 version := 1;  
522:C 5 tps := 1; spm := 1; spt := cpsize div 256;  
523:C 5 for i:=21 to 123 do filler[i]:=0;  
524:C 5 end;  
525:C 4 dummy4 := 0; { clear 250 maint. word }  
526:C 4 { write volume header }  
527:C 4 with unitable^unum do  
528:C 5 BEGIN  
529:C 5 call(tm,fbp(dir),writebytes,vol,sizeof(lvheader),0); goodio;  
530:C 5 for i:=0 to 63 do secbuf[i] := 0;  
531:C 5 call(tm,fbp(dir),writebytes,secbuf,256,256); {clear sector 1}  
532:C 5 END;  
533:C 4 { write end of directory }  
534:C 4 opendir; { clear dentry }  
535:C 4 dentry.ftype := -1;  
536:C 4 writedir(dir,1,dentry); flushdir;  
537:C 4 end; {domakedirectory}  
538:S  
539:D -326 2 (*****  
540:D 2 procedure liftofknd(lt:integer16; var fk:filekind);
```

```
541:C 3 begin  
542:C 3 fk:=untypedfile;  
543:C 3 while (fk>lastfknd) and (efttable^fk)<>lt) do fk:=succ(fk);  
544:C 3 if efttable^fk<>lt then fk:=DATAFILE;  
545:C 3 end;  
546:S  
547:D -326 2 (*****  
548:D 2 procedure cvtdatetime(var fdate:tdate; var date:daterec; var time:timerec);  
549:C 2 begin  
550:C 3 date.year:=fdate[1]*10+fdate[2];  
551:C 3 date.month:=fdate[3]*10+fdate[4];  
552:C 3 date.day:=fdate[5]*10+fdate[6];  
553:C 3 if (date.month=0) or (date.day=0) then  
554:C 4 begin date.year:=0; date.month:=3; date.day:=1; end;  
555:C 3 time.hour:=fdate[7]*10+fdate[8];  
556:C 3 time.minute:=fdate[9]*10+fdate[10];  
557:C 3 time.centisecond:=(fdate[11]*10+fdate[12])*100;  
558:C 3 end;  
559:S  
560:D -326 2 (*****  
561:D 2 procedure doopendirectory(anyvar cat:catentry);  
562:C 2 begin  
563:C 3 opendir; checkvalid;  
564:C 3 with cat do  
565:C 4 begin { volume info }  
566:C 4 cname:=volid;  
567:C 4 cstart:=(vol.dstart+vol.dsize)*256; { start of data area }  
568:C 4 cblocksize:=256; { No. of bytes in allocation unit }  
569:C 4 cpsize:=volsize; { physical size of the volume }  
570:C 4 clsize:=cpsize-cstart; { data space on the medium }  
571:C 4 cextral:=vol.dsize*8; { number of possible files }  
572:C 4 cextra2:=-1; { unused space available }  
573:C 4 cvtdatetime(vol.sdate,clastdate,clasttime); { system date }  
574:C 4 cvtdatetime(vol.ldate,clcreate,clcreate); { date created }  
575:C 4 cinfo:='LIF' level; cinfo[11]:=chr(vol.version+ord('0'));  
576:C 4 end; { volume info }  
577:C 3 end;  
578:S  
579:D -326 2 (*****  
580:D 2 procedure docat(anyvar cat : catarray);  
581:D 2 var  
582:D -12 3 d1, dstart, dnum : integer;  
583:D -13 3 done : boolean;  
584:S  
585:D 3 procedure zerodatetime(var dr:daterec; var tr:timerec);  
586:C 3 begin  
587:C 4 dr.year:=0; dr.month:=3; dr.day:=1;  
588:C 4 tr.hour:=0; tr.minute:=0; tr.centisecond:=0;  
589:C 4 end;  
590:S  
591:C 3 begin {docat}  
592:C 3 opendir; checkvalid;  
593:C 3 dstart:=f.fpos; dnum:=f.fpeof; f.fpeof:=0;  
594:C 3 d1:=1; seek(d1,d1); done:=false;  
595:C 3 while (f.fpeof<dnum) and (not done) do  
596:C 4 begin { get file info }  
597:C 4 read(dir,dentry);  
598:C 4 if dentry.ftype=-1 then done:=true  
599:C 4 else  
600:C 5 { don't show temporary or purged files }
```



```

601:C      5      if (dentry.ftype<>0) and
602:C      6      ((dentry.fdate[3]<>9) or (dentry.fdate[4]<>9)) then
603:C      6      begin { count this entry }
604:C      6      if dstart<=0 then { skip to start index }
605:C      7      begin { report this entry }
606:C      7      fpeof := f.fpeof+1;
607:C      7      with dentry, cat[f.fpeof] do
608:C      8      begin
609:C      8      ifnametostr(fname,cname);
610:C      8      ceft := ftype; liftokkind(ceft,ckind);
611:C      8      cpsize := fsize*256;
612:C      8      if ftype=-5622 (workstation data) then clsize:=extension
613:C      8      else clsize := cpsize;
614:C      8      cstart := fstart*256;
615:C      8      cblocksize := 256;
616:C      8      zerodatetime(ccreatedate,ccreatetime);
617:C      8      cvtdatetime(fdate,clastdate,clasttime);
618:C      8      cextral := extension; cextra2 := volnumber;
619:C      8      if lastvol then cinfo:='' else cinfo := 'continued';
620:C      8      end { with }
621:C      8      end { report this entry }
622:C      7      else dstart := dstart-1;
623:C      6      end; { count this entry }
624:C      4      d1:=d1+1;
625:C      4      if d1>dlast then done := true;
626:C      4      end; { while }
627:C      3      end; { docat }
628:C      3
-326: 2      (*****
630:D      2      function findfile(temporary:boolean; ftypecode:integer16):boolean;
631:D      3      var
632:D      -1 3      found : boolean;
633:D      -12 3      tempname : lifname;
634:D      3
635:C      3      begin {find file}
636:C      3      if not f.fanonymous then strtolifname(f.ftid,tempname);
637:C      3      found:=false; dindex:=1; seek(dir^,dindex);
638:C      3      repeat
639:C      4      read(dir^,dentry);
640:C      4      with dentry do
641:C      5      begin
642:C      5      if ftype=-1 then dindex:=dlast
643:C      5      else
644:C      5      if ftype<>0 then
645:C      5      begin { check this entry }
646:C      5      if f.fanonymous then found:=(dentry.fstart*256 = f.fileid)
647:C      5      else
648:C      5      if (tempname=fname) then
649:C      5      if ((ftypecode=0) or (ftypecode=ftype)) then
650:C      5      begin
651:C      10      if temporary then found:=(fdate[3]=9) and (fdate[4]=9)
652:C      10      else found:=(fdate[3]<>9) or (fdate[4]<>9);
653:C      10      end;
654:C      7      end;
655:C      5      end; { with }
656:C      4      if not found then dindex:=dindex+1;
657:C      4      until (dindex>dlast) or found;
658:C      3      findfile:=found;
659:C      3      end; {find file}
660:C      3

```

```

661:D      -326 2      (*****
662:D      2      procedure purgef;
663:C      3      begin
664:C      3      dentry.ftype := 0; writedir(dir^,dindex,dentry); flushdir;
665:C      3      end;
666:D      3
-326: 2      (*****
668:D      2      procedure dopurgename;
669:C      3      begin
670:C      3      opendir; checkvalid; checkfitle;
671:C      3      if findfile(false,0) then purgef;
672:C      4      else ioreresult := ord(inofile);
673:C      3      end;
674:D      3
-326: 2      (*****
675:D      2      procedure getspace(space:integer; var srec:spacerec);
676:D      3      var
677:D      -3 3      fixed, done, opening : boolean;
678:D      -12 3      lastused, lastopening : integer;
679:D      -16 3      dataspace : integer;
680:D      -48 3      mostavail, nextavail : spacerec;
681:D      3
682:D      3      procedure shuffle;
683:D      4      var
684:D      4      tempentry : direntry;
685:D      -32 4      increment, here, hole : integer;
686:D      -44 4
687:D      4
688:C      4      begin { move directory entries to open a required space }
689:C      4      here := mostavail.here;
690:C      4      hole := mostavail.hole;
691:C      4      if here<hole then
692:C      5      begin
693:C      5      if hole<dlast then
694:C      6      begin { move logical eod if required }
695:C      6      readdrir(dir^,hole,tempentry);
696:C      6      if tempentry.ftype=-1 then
697:C      7      begin
698:C      7      writedir(dir^,hole + 1,tempentry);
699:C      7      anychange := true;
700:C      7      end;
701:C      6      end;
702:C      5      increment := -1;
703:C      5      end
704:C      5      else
705:C      5      increment := 1;
706:C      5      while hole<here do
707:C      5      begin
708:C      5      readdrir(dir^,hole+increment,tempentry);
709:C      5      writedir(dir^,hole,tempentry); anychange := true;
710:C      5      hole := hole + increment;
711:C      5      end;
712:C      4      tempentry.ftype:=0;
713:C      4      writedir(dir^,here,tempentry); anychange := true;
714:C      4      end; { shuffle }
715:D      3
716:D      3      procedure allocate(var srec : spacerec; eod : boolean);
717:C      4      begin
718:C      4      srec.ssize := dataspace;
719:C      4      srec.sstart := lastused + 1;
720:C      4      if eod then

```



```

721:C      5      begin { end of directory allocation }
722:C      5      if opening then srec.here := lastopening
723:C      5      else srec.here := dindex;
724:C      5      srec.hole := srec.here;
725:C      5      end
726:C      5      else
727:C      5      begin { middle of directory allocation }
728:C      5      if opening then srec.here := lastopening
729:C      6      else
730:C      6      if lastopening>0 then srec.here := dindex-1
731:C      7      else srec.here := dindex;
732:C      5      srec.hole := lastopening;
733:C      5      end;
734:C      4      end; { allocate }
735:S      3
736:D      3      procedure checkspace(eod : boolean);
737:D      4      var
738:D      4      temp      : integer;
739:D      4      check2     : boolean;
740:C      4      begin
741:C      4      if fixed and (dataspace>=space) then
742:C      5      begin { fixed space check }
743:C      5      if (mostavail.ssize=0) { no space yet } or
744:C      6      {dataspace=space} { exact fit } or
745:C      6      { this space is bigger than previous good fit }
746:C      6      { ((dataspace-space)>(mostavail.ssize-space)) and
747:C      6      { (mostavail.ssize<space) } then allocate(mostavail,eod);
748:C      5      if mostavail.hole>0 then done := true;
749:C      5      end; { fixed space check }
750:S      4
751:C      4      if not fixed then
752:C      5      begin { biggest or 2nd biggest }
753:C      5      check2 := true;
754:C      5      if (dataspace>=mostavail.ssize) then
755:C      6      begin { check biggest space }
756:C      6      if (dataspace>mostavail.ssize) then
757:C      7      begin { new biggest space }
758:C      7      nextavail := mostavail; { demote to second biggest }
759:C      7      allocate(mostavail,eod);
760:C      7      check2 := false;
761:C      7      end
762:C      7      else
763:C      7      if not eod then
764:C      8      begin { same size space }
765:C      8      if opening then temp := lastopening
766:C      9      else
767:C      9      if lastopening>0 then temp := dindex -1
768:C      10     else temp := dindex;
769:C      10     if (abs(temp-lastopening)<=abs(mostavail.here-mostavail.hole)) or
770:C      9     (lastopening=0) then
771:C      9     with mostavail do
772:C      10     begin { this causes shorter shuffle }
773:C      10     ssize := dataspace;
774:C      10     sstart := lastused + 1;
775:C      10     here := temp;
776:C      10     hole := lastopening;
777:C      10     end;
778:C      8     end; { same size space }
779:C      6     end; { check biggest space }
780:C      5     if check2 then

```

```

781:C      6      if (dataspace>=nextavail.ssize) and (space<0) then
782:C      7      begin { check 2nd biggest space }
783:C      7      if (dataspace>nextavail.ssize) then allocate(nextavail,eod)
784:C      8      else
785:C      8      if not eod then
786:C      9      begin { same size space }
787:C      9      if opening then temp := lastopening
788:C      10     else
789:C      10     if lastopening>0 then temp := dindex -1
790:C      11     else temp := dindex;
791:C      9     if (abs(temp-lastopening)<=abs(nextavail.here-nextavail.hole)) or
792:C      10     (lastopening=0) then
793:C      10     with nextavail do
794:C      11     begin { this causes shorter shuffle }
795:C      11     ssize := dataspace;
796:C      11     sstart := lastused + 1;
797:C      11     here := temp;
798:C      11     hole := lastopening;
799:C      11     end;
800:C      9     end;
801:C      7     end; { same size space }
802:C      5     end; { biggest or 2nd biggest }
803:C      4     end; {checkspace}
804:S      3
805:D      3      procedure checkentry;
806:C      4      begin { checkentry }
807:C      4      if dentry.ftype=-1 then
808:C      5      begin { logical end of directory }
809:C      5      dataspace := (volsize div 256) - lastused - 1;
810:C      5      if dataspace>0 then checkspace(true);{check space at end of directory}
811:C      5      { set lastopening so outer proc. won't think directory is full }
812:C      5      if lastopening=0 then lastopening := dindex;
813:C      5      if (mostavail.hole=0) and (mostavail.sstart>0) then
814:C      6      mostavail.hole := lastopening;
815:C      6      if (nextavail.hole=0) and (nextavail.sstart>0) then
816:C      6      nextavail.hole := lastopening;
817:C      5      done := true; lastused := (volsize div 256) + 1;
818:C      5      end
819:C      5      else
820:C      5      if dentry.ftype=0 then
821:C      6      begin { hole in the directory }
822:C      6      if not opening then
823:C      7      begin
824:C      7      opening := true;
825:C      7      lastopening := dindex;
826:C      7      if mostavail.sstart>0 then
827:C      8      with mostavail do
828:C      9      begin { adjust fixed/biggest space }
829:C      9      if hole=0 then hole := lastopening
830:C      10     else
831:C      10     if (abs(hole-here)>abs(lastopening-here)) then
832:C      11     begin { hole changed direction from entry }
833:C      11     here := here + 1;
834:C      11     hole := lastopening;
835:C      11     end;
836:C      9     if fixed then done := true;
837:C      9     end; { with }
838:S      3
839:C      7      if (space<0) and (nextavail.sstart>0) then
840:C      8      with nextavail do

```



```

841:C      9      begin      ( adjust second biggest space )
842:C      9      if hole=0 then hole := lastopening
843:C      10     else
844:C      10         if (abs(hole-here)>abs(lastopening-here)) then
845:C      11         begin      ( hole changed direction from entry )
846:C      11             here := here + 1;
847:C      11             hole := lastopening;
848:C      11         end;
849:C      9      end;
850:S
851:C      7      end;
852:C      6      end      ( hole in the directory )
853:C      6      else
854:C      6      begin      ( have file entry )
855:C      6          dataspace := dentry.fstart - lastused - 1;
856:C      6          if dataspace>0 then checkspace(false);
857:C      6          ( if no dataspace yet, move lastopening to end of series )
858:C      6          if (mostavail.sstart=0) and opening then lastopening := dindex - 1;
859:C      6          opening := false;
860:C      6          lastused := dentry.fstart + dentry.fsize - 1;
861:C      6      end;
862:C      4      end; (checkentry)
863:S
864:C      3      begin (getspace)
865:C      3          dindex := 1;
866:C      3          seek(dir^,dindex);
867:C      3          if space>0 then begin fixed:=true; space:=(space+255) div 256; end
868:C      4          else fixed:=false;
869:C      4          lastused := dindex - 1;      lastopening := 0;
870:C      3          mostavail.sstart := 0;      nextavail.sstart := 0;
871:C      3          mostavail.ssize := 0;      nextavail.ssize := 0;
872:C      3          mostavail.hole := 0;      nextavail.hole := 0;
873:C      3          mostavail.here := 0;      nextavail.here := 0;
874:C      3          done := false;      opening := false;
875:C      3          repeat
876:C      4              read(dir^,dentry);
877:C      4              checkentry;
878:C      4              dindex := dindex + 1;
879:C      4              if not done then done := dindex>dlast;
880:C      4          until done;
881:S
882:C      3      if lastopening=0 then badio(idirfull)
883:C      4      else
884:C      4      begin      ( have at least one directory opening )
885:C      4          if (mostavail.sstart=0) or not fixed then
886:C      5          begin
887:C      5              dataspace := (vsize div 256) - lastused - 1;
888:C      5              if dataspace>0 then
889:C      6              begin
890:C      6                  dindex := dlast + 1; (insurance policy)
891:C      6                  if (not fixed and (dataspace>mostavail.ssize)) or
892:C      6                  ((space<0) and (dataspace>nextavail.ssize)) or
893:C      6                  (fixed and (dataspace>=space)) then checkspace(false);
894:C      6              end;
895:C      5          end;
896:C      4          if mostavail.sstart=0 then badio(inoroom);
897:S
898:C      4          if fixed then mostavail.ssize := space
899:C      4          else
900:C      4          with mostavail do

```

```

901:C      6      begin      ( final decision for non fixed space )
902:C      6          watch for [*] type allocation
903:C      6          { biggest of [1/2 biggest or 2nd biggest] } ( include any odd sector )
904:C      6          if space<0 then ssize:=(ssize div 2) + (ssize mod 2);
905:C      6          if nextavail.ssize>ssize then mostavail:=nextavail;
906:C      6      end;
907:C      4      shuffle; ( align dataspace and directory entry )
908:C      4      srec := mostavail;
909:C      4      end;
910:C      3      end; (getspace)
911:S
912:D      -326 2 {*****}
913:D      2      procedure finishfib;
914:C      2      begin
915:C      2          f.fileid := dentry.fstart*256;
916:C      2          f.fpeof := dentry.fsize*256;
917:C      2          if f.fisnew then f.fleof := 0
918:C      2          else f.fleof := f.fpeof;
919:C      2          f.fmodified := f.fisnew;
920:C      2          if not f.fbuffered then f.am := amtable^[UNTYPEDFILE]
921:C      2          else
922:C      2          if f.fistextvar then f.am := amtable^[f.fkind]
923:C      2          else f.am := amtable^[datafile];
924:C      2      end; (finishfib)
925:S
926:D      -326 2 {*****}
927:D      2      procedure opennew;
928:D      2      var
929:D      -16      space      : spacerec;
930:C      2      begin
931:C      2          getspace(f.fpos,space);
932:C      2          dindex := space.here;
933:C      2          ( fill in the lif directory entry )
934:C      2          ( file name )
935:C      2          if f.fanonymous then dentry.fname := 'anonymous '
936:C      2          else strtolifname(f.ftid,dentry.fname);
937:C      2          dentry.ftype := f.ftid;
938:C      2          dentry.fstart := space.sstart;
939:C      2          dentry.fsize := space.ssize;
940:C      2          setdate(dentry.fdate);
941:C      2          dentry.fdate[3] := 9; dentry.fdate[4] := 9; ( mark as temporary )
942:C      2          dentry.lastvol := true;
943:C      2          dentry.volnumber := 1;
944:C      2          dentry.extension := 0;
945:C      2          writedir(dir^,dindex,dentry); (write it out) flushdir;
946:C      2          ( finish the file fib )
947:C      2          finishfib;
948:C      2      end; (opennew)
949:S
950:D      -326 2 {*****}
951:D      2      procedure openold;
952:D      -2      var fk:filekind;
953:C      2      begin
954:C      2          if tokind(dentry.ftype,fk); f.fkind:=fk;
955:C      2          finishfib;
956:C      2          with dentry do
957:C      2          begin
958:C      2              f.startaddress := 0;
959:C      2              f.ft := ftype;
960:C      2              if ftype=fttable^[datafile] then f.fleof := extension

```



```
961:C      5      else f.startaddress := extension;
962:C      4
963:C      3      end;
964:S
965:O -326 2 (*****
966:D      2      procedure openf;
967:C      3      begin
968:C      3          opendir; checkvalid;
969:C      3          if not (f.fanonymous and f.fisnew) then checkfitle;
970:C      3          if f.fisnew then
971:C      4              begin { new temp file }
972:C      4                  if not mediavalid then cleandir;
973:C      4                  if f.fanonymous then opennew
974:C      5                  else
975:C      5                      if findfile(f.fisnew,0) then ioresult:=ord(idupfile)
976:C      6                      else opennew;
977:C      4                  end
978:C      4                  else { existing permanent file }
979:C      4                  begin
980:C      4                      if findfile(f.fisnew,0) then
981:C      5                          begin
982:C      5                              openold;
983:C      5                              if not mediavalid then cleandir;
984:C      5                              end
985:C      5                          else ioresult:=ord(inofile);
986:C      4                          end;
987:C      3                      end; { openf }
988:S
989:O -326 2 (*****
990:D      2      procedure closef;
991:D      3      var
992:C      -4      temp : integer;
993:S
994:C      3      begin
995:C      3          if f.fisnew then
996:C      4              begin { purge old file }
997:C      4                  temp:=dindex;
998:C      4                  if findfile(false,0) then purgef;
999:C      4                  dindex:=temp; readdir(dir^,dindex,dentry);
1000:C      4                  end;
1001:C      3          if f.modified then
1002:C      4              with dentry do
1003:C      5                  begin { rewrite the directory entry }
1004:C      5                      if ftype=fttable^[datafile] then extension := f.fleof
1005:C      6                      else extension := f.fstartaddress;
1006:C      5                      temp:=(f.fleof+255) div 256;
1007:C      5                      if temp<fsize then fsize:=temp;
1008:C      5                      setdate(fdate);
1009:C      5                      writedir(dir^,dindex,dentry); flushdir;
1010:C      5                  end;
1011:C      3          end;
1012:S
1013:O -326 2 (*****
1014:D      2      procedure stretchf;
1015:D      3      var
1016:C      -2      found, eod : boolean;
1017:O -18 3      tempindex, filestart, dataspace, reqsize : integer;
1018:S
1019:C      3      begin {stretchf}
1020:C      3          tempindex := dindex; found := false; eod := false;
```

```
1021:C      3      filestart := dentry.fstart;
1022:C      3      reqsize := (f.fpos + 255) div 256; { round requested size 25jan83}
1023:C      3      if reqsize>dentry.fsize then
1024:C      4          begin
1025:C      4              while (not found) and (not eod) do
1026:C      5                  with dentry do
1027:C      6                      begin
1028:C      6                          tempindex := tempindex + 1;
1029:C      6                          if tempindex>dlast then eod := true
1030:C      7                          else
1031:C      7                              begin
1032:C      7                                  readdir(dir^,tempindex,dentry);
1033:C      7                                  if ftype=-1 then eod := true
1034:C      8                                  else
1035:C      8                                      if ftype<>0 then
1036:C      9                                          begin
1037:C      9                                              found := true;
1038:C      9                                              dataspace := dentry.fstart - filestart;
1039:C      9                                          end;
1040:C      7                                  end;
1041:C      6                              end; { while with }
1042:C      4                          if eod then
1043:C      5                              begin { dataspace is from beginning of file to end of volume }
1044:C      5                                  found:=true; {25jan83}
1045:C      5                                  dataspace := (volsize div 256) - filestart; {25jan83}
1046:C      5                              end;
1047:C      4                          if found then
1048:C      5                              if dataspace>=reqsize then {25jan83}
1049:C      6                                  begin { will stretch }
1050:C      6                                      readdir(dir^,dindex,dentry);
1051:C      6                                      { allow requested space + half of excess space 25jan83 }
1052:C      6                                      dentry.fsize := (reqsize + dataspace) div 2; {25jan83}
1053:C      6                                      writedir(dir^,dindex,dentry); flushdir;
1054:C      6                                      f.fpos := dentry.fsize + 256;
1055:C      6                                      end; { will stretch }
1056:C      4                              end;
1057:C      3                          end; { stretchf }
1058:S
1059:O -326 2 (*****
1060:D      2      procedure changefname(anyvar n:string255);
1061:D      3      var
1062:C      -4      tempindex : integer;
1063:D      -5      ok : boolean;
1064:S
1065:C      3      begin
1066:C      3          if f.fanonymous then badio(ibadrequest);
1067:C      3          opendir; checkvalid; checkfitle;
1068:C      3          { find the original (permanent file) }
1069:C      3          if not findfile(false,0) then badio(inofile);
1070:C      3          { change the name }
1071:C      3          tempindex := dindex;
1072:C      3          if (strlen(n)=0) or (strlen(n)>tidleng) then badio(ibadtitle);
1073:C      3          f.ftid := n;
1074:C      3          if findfile(false,0) then badio(idupfile)
1075:C      4          else
1076:C      4              begin
1077:C      4                  strtolifname(f.ftid,dentry.fname);
1078:C      4                  writedir(dir^,tempindex,dentry); flushdir;
1079:C      4              end
1080:C      4          end; { changefname }
```



```

1081:S
1082:D -326 2 {*****}
1083:D 2 {*****}
1084:C 2 {*****}
1085:C 2 {*****}
1086:C 2 {*****}
1087:C 2 {*****}
1088:C 2 {*****}
1089:C 2 {*****}
1090:C 2 {*****}
1091:C 2 {*****}
1092:C 2 {*****}
1093:C 2 {*****}
1094:C 2 {*****}
1095:C 2 {*****}
1096:C 2 {*****}
1097:C 2 {*****}
1098:C 2 {*****}
1099:C 2 {*****}
1100:C 2 {*****}
1101:C 2 {*****}
1102:C 2 {*****}
1103:S
1104:D -326 2 {*****}
1105:D -326 2 {*****}
1106:C 2 {*****}
1107:C 2 {*****}
1108:C 2 {*****}
1109:C 2 {*****}
1110:C 2 {*****}
1111:C 2 {*****}
1112:C 2 {*****}
1113:S
1114:D -326 2 {*****}
1115:D -326 2 {*****}
1116:C 2 {*****}
1117:C 2 {*****}
1118:C 2 {*****}
1119:C 2 {*****}
1120:C 2 {*****}
1121:C 2 {*****}
1122:C 2 {*****}
1123:C 2 {*****}
1124:C 2 {*****}
1125:C 2 {*****}
1126:C 2 {*****}
1127:C 2 {*****}
1128:C 2 {*****}
1129:C 2 {*****}
1130:C 2 {*****}
1131:C 2 {*****}
1132:C 2 {*****}
1133:C 2 {*****}
1134:C 2 {*****}
1135:C 2 {*****}
1136:C 2 {*****}
1137:C 2 {*****}
1138:C 2 {*****}
1139:C 2 {*****}
1140:C 2 {*****}

```

```

1141:C 5 strttopac(f,6,vol.volname,true;
1142:C 5 with unitable^[unum] do
1143:C 5 begin
1144:C 5 call(tm,fibp(dir),writebytes(vol,sizeof(lvheader),0);
1145:C 5 goodio; anychange := true;
1146:C 5 end;
1147:C 5 end;
1148:C 4 purgename : dopurgename;
1149:C 4 getvolumedate: getsdate(f);
1150:C 4 setvolumedate: setsdate(f);
1151:C 4 crunch : crunchv;
1152:C 4 catalog : docat(f.fwindow^);
1153:C 4 opendir: doopendirectory(f.fwindow^);
1154:C 4 closedirectory: begin end;
1155:C 4 makedirectory: domakedirectory(f.fwindow^);
1156:C 4 openunit,
1157:C 4 openvolume : begin
1158:C 4 cleanup;
1159:C 4 unblockeddram(f,unum,request);
1160:C 4 mediavalid := unitable^[unum].umediavalid;
1161:C 4 end;
1162:C 4 setunitprefix: if strlen(f.ftitle)>0 then badio(ibadtitle);
1163:C 4 stripname : begin
1164:C 4 checkftitle; setstrlen(f.ftitle,0);
1165:C 4 strtolifname(f.ftid,dentry.fname);
1166:C 4 end;
1167:C 4 otherwise
1168:C 4 ioresult := ord(ibadrequest);
1169:C 4 end; { case request }
1170:C 3 if anychange then flushdir;
1171:C 3 cleanup; { fix umediavalid and ureportchange }
1172:C 3 recover
1173:C 3 begin
1174:C 3 cleanup; { fix umediavalid and ureportchange }
1175:C 3 if (escapecode<0) and (escapecode<-10) then
1176:C 4 begin lockdown; escape(escapecode); end;
1177:C 3 end;
1178:C 2 lockdown;
1179:C 2 end;
1180:C 1 procedure installlifdam;
1181:C 1 begin
1182:C 1 if dir=nil then new(dir);
1183:C 1 end;
1184:C 1 end; { module }
1185:D 1 import lifmodule,loader;
1186:C 1 begin { instlifdam }
1187:C 1 installlifdam;
1188:C 1 markuser;
1189:C 1 end. { rev 2.2x1 }

```

No errors. No warnings.

***** Nonstandard language features enabled *****

LOCKMOD

Description

LOCKMOD provides the function: LOCK and the procedures: UNLOCK and WAITFORLOCK for the sharing of SRM files opened with the LOCKABLE option as the third parameter.

Usage

Only useful with SRM (Shared Resource Managment).

Notes

Supplied in SYSVOL:LIBRARY.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1983.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S     RESTRICTED RIGHTS LEGEND
11:S
12:S     Use, duplication, or disclosure by the Government
13:S     is subject to restrictions as set forth in
14:S     paragraph (b) (3) (B) of the Rights in Technical
15:S     Data and Computer Software clause in
16:S     DARF 7-104.9(a).
17:S
18:S     HEWLETT-PACKARD COMPANY
19:D     0 Fort Collins, Colorado      *)
20:S
21:S
22:S     0 $copyright 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'S
23:S
24:D     0 $sysprogs
25:D     0 module lockmodule;
26:S
27:D     1 import sysglobals;
28:D     1 export
29:S
30:D     1 function lock      (anyvar f : fib) : boolean;
31:D     1 procedure unlock  (anyvar f : fib);
32:D     1 procedure waitforlock (anyvar f : fib);
33:S
34:D     1 implement
35:S
36:D     1 (*****
37:D     1 function lock      (anyvar f : fib) : boolean;
38:D     2 begin
39:D     2   ioresult := ord(inoerror);
40:D     2   with f do
41:D     3     if not flockable then
42:D     4       ioresult := ord(inotlockable)
43:D     4     else
44:D     4       if flocked then
45:D     5         ioresult := ord(ifilelocked)
46:D     5       else
47:D     5         begin
48:D     6           fwaitonlock := false;
49:D     6           call(unitable^[funit].dam,f,funit,lockfile);
50:D     6         end;
51:D     2   if ioresult = ord(inoerror) then
52:D     3     lock := true
53:D     3   else
54:D     3     begin
55:D     4       lock := false;
56:D     4       if ioresult <> ord(ifilelocked) then
57:D     5         escape(-10);
58:D     3     end;
59:D     2 end;
60:S

```

```

61:D     1 (*****
62:D     1 procedure unlock  (anyvar f : fib);
63:D     2 begin
64:D     2   ioresult := ord(inoerror);
65:D     2   with f do
66:D     3     if not flockable then
67:D     4       ioresult := ord(inotlockable)
68:D     4     else
69:D     4       if not flocked then
70:D     5         ioresult := ord(ifileunlocked)
71:D     5       else
72:D     5         call(unitable^[funit].dam,f,funit,unlockfile);
73:D     2   if ioresult <> ord(inoerror) then
74:D     3     escape(-10);
75:D     2 end;
76:S
77:D     1 (*****
78:D     1 procedure waitforlock (anyvar f : fib);
79:D     2 begin
80:D     2   ioresult := ord(inoerror);
81:D     2   with f do
82:D     3     if not flockable then
83:D     4       ioresult := ord(inotlockable)
84:D     4     else
85:D     4       if flocked then
86:D     5         ioresult := ord(ifilelocked)
87:D     5       else
88:D     5         begin
89:D     6           fwaitonlock := true;
90:D     6           call(unitable^[funit].dam,f,funit,lockfile);
91:D     6         end;
92:D     2   if ioresult <> ord(inoerror) then
93:D     3     escape(-10);
94:D     2 end;
95:S
96:C     1 end. (lockmodule)

```

No errors. No warnings.

***** Nonstandard language features enabled *****

M68KSYS

Description

M68KSYS provides the command interpreter for the Pascal operating system.

Requirements

SYSGLOBALS, FS, LOADER, ASM, MISC, SYSDEVS, and LDR.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*
2:S      (c) Copyright Hewlett-Packard Company, 1984.
3:S      All rights are reserved. Copying or other
4:S      reproduction of this program except for archival
5:S      purposes is prohibited without the prior
6:S      written consent of Hewlett-Packard Company.
7:S
8:S
9:S      RESTRICTED RIGHTS LEGEND
10:S
11:S      Use, duplication, or disclosure by the Government
12:S      is subject to restrictions as set forth in
13:S      paragraph (b) (3) (8) of the Rights in Technical
14:S      Data and Computer Software clause in
15:S      DAR 7-104.9(a).
16:S
17:S      HEWLETT-PACKARD COMPANY
18:S      0 Fort Collins, Colorado      *)
19:D
20:D
21:D
22:D
23:D      0 $modcal$
24:D      0 $iocheck off$ $debug off$ $range off$ $ovfcheck off$
25:D      0 $heap_dispose off$
26:D      0 $search 'INITLOAD', 'ASM', 'INIT', 'SYSDEVS'$
27:D
28:D      0 program cmd(input,output,keyboard);
29:D      1
30:D      1 module ci;
31:S
32:D      1 import
33:D      1   sysglobals,fs,loader,asm,misc,sysdevs,ldr;
34:D
35:D      1 export
36:D
37:D
38:D      1 type
39:D      1   sysfiles = (assembler,compiler,editor,file,librarian,library);
40:D      1   sysfilevols = array [sysfiles] of string[6];
41:D      1   sysfilenames = array [sysfiles] of fid;
42:D
43:D      1
44:D      1   inforec = record                                (*FILE INFORMATION*)
45:D      1       errsym,errblk,errnum: integer;              (*ERROR STUFF IN EDIT*)
46:D      1       gotsym,gotcode: boolean;                   (*TITLES ARE MEANINGFUL*)
47:D      1       workfid,symfid,codefid,errfid: fid;         (*PERM&CUR WORKFILES*)
48:D      1       end (*INFOREC*);
49:D
50:D      1   inforecptr = ^inforec;
51:D      1   cmdprocedure = procedure;
52:D      1   cmdprocptr = ^cmdprocedure;
53:S
54:D      -4 1 var streamfib: ^text;                        (*FOR SYSTEM STREAM FILE*)
55:D      -736 1   filename: sysfilenames;
56:D      -740 1   ioresult: integer; (to save previous ioresult)
57:D      -862 1   chainfile: fid;
58:D      -864 1   chaining: (nochain,progchain,streamchain);
59:D      -868 1   userinfo: inforecptr;
60:D      -869 1   versionup: boolean;

```

```

61:D      -874 1   cmdcharhook : cmdprocptr;
62:D      -875 1   ci_idle: boolean;
63:D      -876 1   ci_cmd: char;
64:D      -877 1   keystream: boolean; ( stream file is original file )
65:S
66:D      -877 1   procedure homecursor ;
67:D      -877 1   procedure clearscreen ;
68:D      -877 1   procedure clearline ;
69:D      -877 1   procedure prompt (pl: string80);
70:D      -877 1   function getchar (flushit: boolean): char;
71:D      -877 1   function uppercase (ch: char): char;
72:D      -877 1   procedure chain(filename: fid);
73:D      -877 1   procedure startstream(filename: fid);
74:D      -877 1   function streaming: boolean;
75:S
76:S
77:D      -877 1 procedure systemstartup;
78:S
79:D      -877 1 implement
80:S
81:D      -877 1 type
82:D      -877 1   monthtype = array [0..15] of packed array [1..3] of char;
83:D      -877 1 const
84:D      -877 1   months = monthtype [ '???' , 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
85:D      -877 1       'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec', '???' ,
86:D      -877 1       '???' , '???' ];
87:S
88:D      1 function streaming boolean;
89:D      2 begin with fibp(streamfib) do
90:D      3   streaming := freadable and (fpos < fleof);
91:D      2 end;
92:S
93:D      -122 1 procedure startstream(filename: fid);
94:D      2 begin
95:D      2   chainfile := filename;
96:D      2   chaining := streamchain;
97:D      2   escape(0);
98:D      2 end;
99:S
100:D      -122 1 procedure chain(filename: fid);
101:D      2 begin
102:D      2   chainfile := filename;
103:D      2   filename(chainfile, codefile);
104:D      2   chaining := progchain;
105:D      2   escape(0);
106:D      2 end;
107:S
108:D      1 procedure disptime;
109:D      -4 2 var time: timerec;
110:D      -6 2   date: daterec;
111:D      -14 2   x,y: integer;
112:D      -18 2   second, day: shortint;
113:D      2 begin
114:D      2   second := -1;
115:D      2   day := -1;
116:D      2   setrunlight(chr(idle));
117:D      2   with fibp(gfiles[0]) do
118:D      3     repeat
119:D      4       call(am, fibp(gfiles[0]), unitstatus, gfiles[0], 0, 0);
120:D      4       if fbusy and versionup then

```



```

121:C      5      begin
122:C      5      systime(time);
123:C      5      sysdate(date);
124:C      5      with time, date do
125:C      5      if (centisecond div 100 <> second) or (day <> dayy) then
126:C      7      begin
127:C      7      second := centisecond div 100;
128:C      7      fgotoxy(output, x, y);
129:C      7      fgotoxy(output, 25, 3);
130:C      7      write(hour:2,':',minute:2,':',second:2);
131:C      7      if day <> dayy then
132:C      8      begin
133:C      8      dayy := day;
134:C      8      fgotoxy(output, 25, 2);
135:C      8      writeln(day:2,'-',months[month],'-',year:2);
136:C      8      end;
137:C      7      fgotoxy(output, x, y);
138:C      7      end;
139:C      5      end;
140:C      4      if fbusy and c1_idle then call(kbdwaithook);
141:C      4      until not fbusy or (chaining<>nochain);
142:C      2      c1_idle:=false;
143:C      2      end;
144:S
145:D      1      procedure dummycmdchar;
146:C      2      begin end;
147:S
148:D      1      procedure initdate;
149:D      2      var
150:D      2      thedatetime : datatimerec;
151:D      -10      ltime : timerec;
152:C      2      begin
153:C      2      with thedatetime, date do
154:C      3      begin
155:C      3      sysdate(date);
156:C      3      if (year=0) and (month=3) and (day=1) then
157:C      4      begin
158:C      4      systime(ltime);
159:C      4      time := ltime;
160:C      4      call [unitable^[sysunit].dam, thedatetime, sysunit, getvolumedate];
161:C      4      if ioreult = ord(inoerror) then
162:C      5      begin
163:C      5      setsysdate(date);
164:C      5      with ltime do
165:C      5      if (hour = 0) and (minute = 0) then
166:C      7      begin
167:C      5      end;
168:C      4      end;
169:C      3      end;
170:C      2      end; (*INITDATE*)
171:S
172:D      1      procedure dateset;
173:D      2      var
174:D      -82      gs: string80;
175:D      -83      changed: boolean;
176:D      -88      i: integer;
177:D      -92      clocktime: timerec;
178:D      -94      clockdate: daterec;
179:D      -100      clockdatetime: datatimerec;
180:S

```

```

181:D      2      procedure identify;
182:C      3      begin
183:C      3      clearscreen; writeln(output);
184:C      3      writeln(output);
185:C      3      writeln(output, ' System date is ');
186:C      3      writeln(output, ' Clock time is ');
187:C      3      writeln(output);
188:C      3      writeln(output, ' Workstation ', fsidec);
189:C      3      writeln;
190:C      3      writeln(output, ' Available Global Space ', eglobal-(a5-32768):1, ' bytes');
191:C      3      writeln(output, ' Total Available Memory ',
192:C      3      eglobal-integer(eheap):1, ' bytes');
193:C      3      writeln;
194:C      3      writeln(output, ' System volume: ', syvid, ':');
195:C      3      writeln(output, ' Default volume: ', dkvid, ':');
196:C      3      writeln;
197:C      3      writeln;
198:C      3      writeln('Copyright 1984 Hewlett-Packard Company. ');
199:C      3      writeln('All rights are reserved. Copying or other ');
200:C      3      writeln('reproduction of this program except for archival ');
201:C      3      writeln('purposes is prohibited without the prior ');
202:C      3      writeln('written consent of Hewlett-Packard Company. ');
203:C      3
204:C      3      versionup := true;
205:S
206:C      3      end; {IDENTIFY}
207:S
208:D      2      function readnumericfield
209:D      3      (llimit,hlimit:integer; var field:integer): boolean;
210:D      3      label 1;
211:D      -7      var gotnum: boolean; i: integer; ch: char;
212:C      3      begin gotnum := false; i := 0;
213:C      3      while strien(gs) > 0 do
214:C      4      begin ch := gs[i];
215:C      4      if (ch>='0') and (ch<='9') then
216:C      5      begin gotnum := true;
217:C      5      i := 10*i+ord(ch)-ord('0');
218:C      5      if i>hlimit then i := hlimit+1;
219:C      5      end
220:C      4      else
221:C      5      if gotnum then goto 1;
222:C      4      strdelete(gs,i,1);
223:C      4      end;
224:C      3      if gotnum and (i>=llimit) and (i<=hlimit) then
225:C      4      begin readnumericfield := true; field := i end
226:C      3      else
227:C      4      readnumericfield := false;
228:C      3      end; {READNUMERICFIELD}
229:S
230:D      2      function readmonthabbrev (var monthnumber:integer): boolean;
231:D      3      label 1;
232:D      -4      var s3: packed array[1..3] of char;
233:D      -9      i: integer; ch: char;
234:C      3      begin i := 0; s3 := ' ';
235:C      3      while strien(gs) > 0 do
236:C      4      begin ch := gs[i];
237:C      4      if ch in ['A'..'Z','a'..'z'] then
238:C      5      begin i := i+1;
239:C      5      if (ch>='A') and (ch<='Z') then
240:C      6      ch := chr(ord(ch)-ord('A')+ord('a'));

```



```

241:C      5      if i<4 then s3[i] := ch;
242:C      5      end
243:C      5      else
244:C      5      if i>0 then goto 1;
245:C      4      strdelete(gs,1,1);
246:C      4      end;
247:C      3 1: s3[i] := uppercase(s3[i]); readmonthabbrev := false;
248:C      3      for i := 1 to 12 do
249:C      4      if months[i] = s3 then
250:C      5      begin readmonthabbrev := true; monthnumber := i end;
251:C      5      end; (READMONTHABBREV)
252:C      2
253:C      2 begin (DATESET)
254:C      2 changed := false;
255:C      2 sysdate(clockdate);
256:C      2 systime(clocktime);
257:C      2 identify;
258:C      2 prompt('New system date ? '); disptime;
259:C      2 readln(input,gs);
260:C      2 if strlen(gs)>0 then with clockdate do
261:C      4 if readnumericfield(1,31,i) then
262:C      5 begin
263:C      5 changed := true;
264:C      5 day := i;
265:C      5 if readmonthabbrev(i) then month := i;
266:C      5 if readnumericfield(0,99,i) then year := i;
267:C      5 setsysdate(clockdate);
268:C      5 end;
269:C      2 prompt('New system clock time ? '); disptime;
270:C      2 readln(input,gs);
271:C      2 if strlen(gs)>0 then with clocktime do
272:C      4 if readnumericfield(0,23,i) then
273:C      5 begin
274:C      5 changed := true;
275:C      5 hour := i;
276:C      5 if readnumericfield(0,59,i) then minute := i
277:C      5 else minute := 0;
278:C      5 if readnumericfield(0,59,i) then centisecond := i*100
279:C      5 else centisecond := 0;
280:C      5 setsysime(clocktime);
281:C      5 end;
282:C      2 if changed then
283:C      3 with clockdatetime do
284:C      4 begin
285:C      4 date := clockdate;
286:C      4 time := clocktime;
287:C      4 call (unitable^[sysunit].dam, clockdatetime, sysunit, setvolumedate);
288:C      4 identify;
289:C      4 end;
290:C      2 end; (DATESET)
291:C      2
292:D      1 procedure disableusersrs;
293:D      2 var i:integer;
294:D      2 begin
295:D      2 call(cleariohook);
296:D      2 interrupttable:=perminttable;
297:D      2 end;
298:D      2
299:D      1 function uppercase (CH: CHAR): CHAR;
300:C      2 begin

```

```

301:C      2      if (ch>='a') and (ch<='z')
302:C      3      then uppercase := chr(ord(ch)-32)
303:C      3      else uppercase := ch
304:C      3      end;
305:C      3
306:D      1 procedure streamdvr(fp: fibp; request: amrequesttype; anyvar buffer: window;
307:D      2      bufsize, position: integer);
308:D      2 type str = record s: string255 end;
309:D      2 strptr = ^str;
310:D      2
311:D      -4 2 var buf: charptr;
312:D      -5 2 c: char;
313:D      -8 2 i: shortint;
314:D      2
315:D      2 procedure checkctrl(var c:char);
316:D      3 var
317:D      -1 3 eoln: boolean;
318:D      3 begin { check for control chars in keystream files }
319:D      3 if (c=chr(255)) and keystream then
320:D      4 begin
321:D      4 eoln:=eoln(streamfib^);
322:C      4 read(streamfib^,c);
323:C      4 if eoln then c:=chr(13);
324:C      4 c:=chr(ord(c) mod 32);
325:C      4 end;
326:C      3 end; { checkctrl }
327:D      3
328:D      2 procedure closedown;
329:D      3 begin
330:D      3 if keystream then close(streamfib^); { if keystream keep the file }
331:C      4 else close(streamfib^, 'PURGE');
332:C      3 with fp^ do
333:C      4 begin
334:C      4 am := serialtextamhook;
335:C      4 call(am, fp, request, buf^, bufsize, position);
336:C      4 end;
337:C      3 end;
338:D      3
339:C      2 begin (STREAMDVR)
340:C      2 ioresult := ord(inoerror);
341:C      2 buf := addr(buffer);
342:C      2 if eof(streamfib^) then closedown
343:C      3 else
344:C      3 with fp^, unitable^[funit] do
345:C      4 case request of
346:C      5 readbytes: while bufsize > 0 do
347:C      6 begin
348:C      6 eoln := eoln(streamfib^);
349:C      6 read(streamfib^, buf^); checkctrl(buf^);
350:C      6 if uisinteractive then
351:C      7 if feoln then call(tm, fp, writeeol, buf^, 1, 0)
352:C      7 else call(tm, fp, writebytes, buf^, 1, 0); (echo)
353:C      6 bufsize := bufsize - 1;
354:C      6 if bufsize > 0 then
355:C      7 begin
356:C      7 buf := addr(buf^, 1);
357:C      7 if eof(streamfib^) then begin
358:C      8 closedown;
359:C      8 bufsize := 0;
360:C      8 end;

```



```

361:C      7      end;
362:C      6      end;
363:C      5      readtoeol: with strptr(buf)^ do
364:C      6      begin
365:C      6      setstrlen(s, bufsize);
366:C      6      i := 0;
367:C      6      while (i < bufsize) and not eoln(streamfib^) do
368:C      7      begin i := i + 1; read(streamfib^, s[i]); checkctrl(s[i]); end;
369:C      6      setstrlen(s, i);
370:C      6      {note there is no need to echo, since readtoeol isn't used interactively}
371:C      6      if i < bufsize then
372:C      7      if eof(streamfib^) then
373:C      8      begin
374:C      8      buf := addr(s[i]);
375:C      8      c := s[i];
376:C      8      bufsize := bufsize - i;
377:C      8      closedown;
378:C      8      setstrlen(s, i+ord(s[i]));
379:C      8      s[i] := c;
380:C      8      end;
381:C      6      end;
382:C      6      unitstatus: fbusy := false;
383:C      6      otherwise call(tm, fp, request, buffer, bufsize, position);
384:C      5      end;
385:C      2      end; {STREAMDVR}
386:S
387:D      -82      1      procedure streamopen(sfile: string80; report:boolean);
388:D      -82      2      const
389:D      -82      2      parindex = ['0'..'9','A'..'Z'];
390:D      -82      2      type
391:D      -82      2      stringptr = ^string255;
392:D      -82      2      var
393:D      -254      2      parptr: array['0'..'Z'] of stringptr;
394:D      -262      2      lastior, i: integer;
395:D      -266      2      heap: ipointer;
396:D      -930      2      f: text;
397:D      -948      2      v: vid;
398:D      -1070      2      t: fid;
399:D      -1074      2      segs: integer;
400:D      -1076      2      fk: filekind;
401:S
402:S
403:D      2      function streamsyntax : boolean;
404:D      3      label
405:D      3      i;
406:D      3      var
407:D      -2      3      c, uc : char;
408:D      -166      3      s, instr : string80;
409:D      -169      3      ciseoln, needc, notendofparms : boolean;
410:S
411:D      3      procedure getc;
412:C      4      begin
413:C      4      ciseoln := eoln(f);
414:C      4      read(f, c);
415:C      4      if strlen(instr) = 80 then strdelete(instr, i, 10);
416:C      4      setstrlen(instr, strlen(instr)+1);
417:C      4      instr[strlen(instr)] := c;
418:C      4      end;
419:S
420:D      3      procedure testioresult;

```

```

421:C      4      begin
422:C      4      if ioresult <> ord(inoerror) then
423:C      5      begin
424:C      5      lastior := ioresult;
425:C      5      writeln('Can't create ', sfile);
426:C      5      printerror(-10, lastior);
427:C      5      goto 1;
428:C      5      end;
429:C      4      end;
430:S
431:C      3      begin {STREAMSYNTAX}
432:C      3      mark(heap);
433:C      3      for uc := '0' to 'Z' do parptr[uc] := nil;
434:S
435:C      3      streamsyntax := false;
436:C      3      notendofparms := true;
437:S
438:C      3      if not keystream then { if keystream then forget this whole operation }
439:C      4      while not eof(f) do
440:C      5      begin
441:S
442:C      5      instr := '';
443:C      5      s := '';
444:C      5      getc;
445:C      5      needc := false;
446:S
447:C      5      if (c = '=') and notendofparms then {parm line}
448:C      6      begin
449:C      6      getc;
450:C      6      uc := uppercase(c);
451:C      6      readln(f, s);
452:C      6      if uc in parindex then
453:C      7      begin
454:C      7      writeln(output, s);
455:C      7      newbytes(parptr[uc], sizeof(string255));
456:C      7      readln(input, parptr[uc]^);
457:C      7      end
458:C      7      else
459:C      7      begin
460:C      7      writeln(output, instr, s);
461:C      7      writeln(output, ' ', strlen(instr)-1
462:C      7      mod syscom^.crinfo.width, '^');
463:C      7      printerror(-24, 0);
464:C      7      goto 1;
465:C      7      end;
466:C      6      end;
467:C      6      {line to process}
468:C      6      begin
469:C      6      if notendofparms then
470:C      7      begin
471:C      7      notendofparms := false;
472:C      7      rewrite(streamfib^, sfile, 'exclusive');
473:C      7      testioresult;
474:C      7      end;
475:S
476:C      6      repeat
477:C      7      if needc then getc;
478:S
479:C      7      if c = chr(255) then {control char}
480:C      8      begin

```



```

481:C      8      getc;
482:C      8      if ciseoln then c := chr(13);
483:C      8      c := chr(ord(c) mod 32);
484:C      8      write(streamfib^,c);
485:C      8      end;
486:C      8      else if c = '@' then          (macro expansion)
487:C      9      begin
488:C      9      getc;
489:C      9      if ciseoln then
490:C      10     begin          (error: no char after @)
491:C      10     writeln(output,instr);
492:C      10     writeln(output,'':strlen(instr)-1
493:C      10     mod syscom^.crtinfo.width,'^');
494:C      10     printerror(-25,0);
495:C      10     goto 1;
496:C      10     end;
497:C      9      uc := uppercase(c);
498:C      9      if uc in parindex then
499:C      10     if parptr[uc] <> nil then write(streamfib^,parptr[uc]^)
500:C      11     else
501:C      11     begin
502:C      11     if not ciseoln then readln(f,s);
503:C      11     writeln(output,instr,s);
504:C      11     writeln(output,'':strlen(instr)-1
505:C      11     mod syscom^.crtinfo.width,'^');
506:C      11     printerror(-25,0);
507:C      11     goto 1;
508:C      11     end;
509:C      11     else          (write char as is)
510:C      10     write(streamfib^,c);
511:C      9      end;
512:C      9      else if ciseoln then          (char is eoln)
513:C      10     begin
514:C      10     if not eof(f) then writeln(streamfib^);
515:C      10     end;
516:C      10     else          (normal char)
517:C      10     write(streamfib^,c);
518:C      10
519:C      7      testioresult;
520:C      7      needc := true;
521:C      7
522:C      7      until ciseoln or eof(f);
523:C      6      end; (line to process)
524:C      5      end; (while not eof(f))
525:C      3      testioresult;
526:C      3      streamsyntax := true;
527:C      3      1: release(heap);
528:C      3      3 end; (STREAMSYNTAX)
529:C
530:C
531:C
532:C
533:C
534:C
535:C
536:C      2 begin (STREAMOPEN)
537:C      2 reset(f,sfile,'shared');          (OPEN THE STREAM FILE)
538:C      2 if ioresult = ord(inoerror) then    (SUCCESSFUL OPEN)
539:C      3 begin          (OPEN THE SYSTEM STREAM FILE)
540:C      3 if scantitle(sfile,v,t,segs,fk) then keystream:=segs<>0

```

```

541:C      4      else keystream:=false;
542:C      3      if not keystream then sfile := '*STREAM';
543:C      3      if streamsyntax then          (SYNTAX AND WRITE TO SYSTEM FILE)
544:C      4      begin          (AVOID HOGGING THE DISK)
545:C      4      if not keystream then close(streamfib^, 'CRUNCH');
546:C      4      reset(streamfib^,sfile,'shared');
547:C      4      fbp(gfiles[0])^am := streamvtr;          (INPUT)
548:C      4      fbp(gfiles[2])^am := streamvtr;          (KEYBOARD)
549:C      4      end;
550:C      4      else close (streamfib^, 'PURGE');          (REMOVE SYSTEM STREAM FILE)
551:C      3      close (f);
552:C      3      end;
553:C      3      else if report then writeln(output,'Can't open file ',sfile);
554:C      2 end; (STREAMOPEN)
555:C
556:C      1 procedure homecursor; begin write(homechar); end;
557:C
558:C      1 procedure clearscreen; begin write(clearscr); versionup := false; end;
559:C
560:C      1 procedure clearline; begin write(cteol ); end;
561:C
562:D      -82 1 procedure prompt (*PL: STRING80*);
563:C      2 begin homecursor; clearline; write(output,pl) end;
564:C
565:D      1 procedure zaptypahead;
566:D      -4 2 var x: integer;
567:C      2 begin call (fbp(gfiles[2])^am, fbp(gfiles[2]), clearunit, x, 0, 0);
568:C      2 reset(input); reset(gfiles[2]^ (KEYBOARD));
569:C      2 end;
570:C
571:D      1 function getchar (flushit: boolean): char;
572:D      -1 2 var ch: char;
573:C      2 begin
574:C      2 if flushit then zaptypahead;
575:C      2 read(input,ch);
576:C      2 getchar := uppercase (ch);
577:C      2 end (*GETCHAR*);
578:C
579:D      1 procedure initfnames;
580:C
581:D      2 const sf = sysfilenames
582:D      2 ['ASSEMBLER','COMPILER','EDITOR','FILER','LIBRARIAN','LIBRARY'];
583:D      2 sysvolname = sysfilevol;
584:D      2 ['ASH','CMP','ACCESS','ACCESS','ACCESS','SY5VOL']; ( js 8/5/83 )
585:C
586:D      -2 2 var f: sysfiles;
587:D      -6 2 find: set of sysfiles;
588:D      -8 2 lunit: unitnum;
589:C
590:D      2 procedure findem(var volume: vid);
591:D      -2 3 var f: sysfiles;
592:D      -668 3 l: file of integer;
593:D      -750 3 ltitle: string80;
594:C      3 begin
595:C      3 for f := assembler to library do
596:C      3 if f in find then
597:C      3 begin
598:C      3 ltitle := volume+':'+sf[f];
599:C      3 reset(l,ltitle,'shared');
600:C      3 if ioresult = ord(inoerror) then

```



```

601:C      6      begin
602:C      6      filename[f] := ltitle;
603:C      6      find := find - [f];
604:C      6      close(l);
605:C      6      end;
606:C      5      end; (findem)
607:C      3
608:S
609:C      2      begin ( initfnames )
610:C      2      find := [assembler..library];
611:C      2      findem(syvid);
612:C      2      lunit := 1;
613:C      2      while (lunit <= maxunit) and (find <> []) do with unitable^[lunit] do
614:C      4      begin
615:C      4      call (dam, uvid, lunit, getvolumename);
616:C      4      if uisblk and (uvid <> '?') and (uvid <> syvid) then findem(uvid);
617:C      4      lunit := lunit+1
618:C      4      end;
619:C      2      for f := assembler to library do
620:C      2      if f in find then filename[f] := sysvolname[f]+'.'+sf[f];
621:C      2      syslibrary := filename[library];
622:C      2      end (*INITFNAMES*);
623:S
624:D      1      procedure initworkfile;
625:D      2      var ltitle: string80;
626:D      2      workfile: file of integer;
627:D      2      begin
628:D      3      with userinfo^ do
629:D      3      begin
630:D      3      errnum := 0; errblk := 0; errsyn := 0; (*INITIALIZE WORK FILES*)
631:C      3      symfid := ''; codefid := ''; workfid := ''; errfid := '';
632:S
633:C      3      ltitle := '*WORK.TEXT';
634:C      3      reset(workfile, ltitle, 'shared');
635:C      3      gotsym := ioresult = ord(ioerror);
636:C      3      if gotsym then symfid := ltitle;
637:C      3      close(workfile);
638:S
639:C      3      ltitle := '*WORK.CODE';
640:C      3      reset(workfile, ltitle, 'shared');
641:C      3      gotcode := ioresult = ord(ioerror);
642:C      3      if gotcode then codefid := ltitle;
643:C      3      close(workfile);
644:C      3      end;
645:C      2      end (*INITWORKFILE*);
646:S
647:D      1      procedure updatesysunit;
648:D      2      begin
649:D      2      initdate;
650:D      2      initfnames;
651:D      2      initworkfile;
652:D      2      end;
653:S
654:D      1      procedure whatfiles;
655:S
656:D      -14      2      var e: string[12];
657:D      -16      2      f: sysfiles;
658:D      -16      2      update: boolean; c: char;
659:D      -22      2      i: integer;
660:S

```

```

661:D      2      procedure edit(f: sysfiles);
662:D      -122      3      var s: fid;
663:C      3      begin
664:C      3      fgotoxy(output, 12, 3+ord(f)); write(cteol); readln(s);
665:C      3      ftxname(s, codefid);
666:C      3      if s <> '' then filename[f] := s;
667:C      3      fgotoxy(output, 12, 3+ord(f)); write(filename[f], cteol);
668:C      3      end;
669:S
670:D      2      procedure volname(sysvol: boolean);
671:D      -4      3      var i: integer;
672:D      -126      3      s: fid;
673:C      3      begin
674:C      3      fgotoxy(output, 19, 11-ord(sysvol)); write(cteol); readln(s);
675:C      3      zapspaces(s);
676:C      3      if s <> '' then
677:C      4      begin
678:C      4      if sysvol then doprefix(s, syvid, sysunit, true)
679:C      4      else doprefix(s, dkvid, i, false);
680:C      4      if ioresult <> ord(ioerror) then
681:C      5      begin
682:C      5      getioerrmsg(s, ioresult); fgotoxy(output, 0, 13);
683:C      5      writeln(belchar, s, cteol);
684:C      5      end
685:C      4      else if sysvol then begin updatesysunit; update := true; end;
686:C      4      end;
687:C      3      fgotoxy(output, 19, 11-ord(sysvol));
688:C      3      if sysvol then write(syvid) else write(dkvid);
689:C      3      write(' ', cteol);
690:C      3      end;
691:D      -22      2
692:C      2      begin ( whatfiles )
693:C      2      filename[library] := syslibrary; update := true;
694:C      2      repeat
695:C      3      if update then
696:C      4      begin
697:C      4      page;
698:C      4      writeln; writeln; writeln;
699:C      4      for f := assembler to library do
700:C      5      begin
701:C      5      e := '';
702:C      5      strwrite(e, i, i, f);
703:C      5      writeln(e, '12-strlen(e), filename[f]);
704:C      5      end;
705:C      4      writeln;
706:C      4      writeln('* System volume: ', syvid, ':');
707:C      4      writeln(' Default volume: ', dkvid, ':');
708:C      4      update := false;
709:C      4      end;
710:C      3      writeln(homechar, 'Assembler Compiler Editor Filer Librarian', cteol);
711:C      3      write ( 'Library System volume Default volume Quit ', cteol);
712:C      3      c := getchar(false);
713:C      3      fgotoxy(output, 0, 13); write(cteol);
714:C      3      case c of
715:C      4      'A': edit(assembler);
716:C      4      'B': edit(library);
717:C      4      'C': edit(compiler);
718:C      4      'D': volname(false); (prefix,)
719:C      4      'E': edit(editor);
720:C      4

```



```

721:C      4      'F': edit(filer);
722:C      4      'L': edit(librarian);
723:C      4      'S': volname(true);      {sysvol;}
724:C      4      'Q': ;
725:C      4      otherwise write(belchar);
726:C      4      end;
727:C      3      until c = 'Q';
728:C      2      syslibrary := filename[library];
729:C      2      end; { whatfiles }
730:S
731:D      1      procedure ramdriver(fp: fibp; request: amrequesttype; anyvar buffer: window;
732:D      2      length, position: integer);
733:C      2      begin
734:C      3      with fp^, unitable^[funit] do
735:C      4      case request of
736:C      5      flush:
737:C      6      writebytes: if length > 0 then
738:C      7      fastmove(addr(buffer), ipointer(byteoffset + position + fileid), length);
739:C      4      readbytes: if length > 0 then
740:C      5      fastmove(ipointer(byteoffset + position + fileid), addr(buffer), length);
741:C      4      otherwise ioresult := ord(ibadrequest);
742:C      4      end;
743:C      2      end; { ramdriver }
744:S
745:D      1      function getvalue(low,high: integer; var value: integer;
746:D      2      numsign, opt: boolean): boolean;
747:D      2      var s: string80;
748:D      2      i: integer;
749:C      2      begin
750:C      3      readln(s);
751:C      4      if (strlen(s)=0) and opt then s := '0';
752:C      2      strread(s,i,value);
753:C      3      if ioresult<>ord(inoerror) then
754:C      4      if i <= strlen(s) then
755:C      5      if (s[i]='#') and numsign then
756:C      6      begin
757:C      7      strread(s,i+1,value);
758:C      5      if ioresult<>0 then writeln('#7'integer required')
759:C      6      end
760:C      5      else writeln('#7'integer required');
761:C      3      if ioresult=ord(inoerror) then
762:C      4      if (value<low) or (value>high) then
763:C      5      begin
764:C      6      writeln('#7'value must be between ',
765:C      7      low:1, ' and ', high:1);
766:C      4      ioresult := ord(ibadformat);
767:C      4      end;
768:C      2      getvalue := ioresult=0;
769:C      2      end; { getvalue }
770:S
771:D      1      procedure makeramvol;
772:S
773:D      2      const zero = direntry[
774:D      2      dfirstblk: 0,      dlastblk: 6,
775:D      2      dfkind: untypedfile,
776:D      2      dvid: 'RAM',      deovblk: 6,
777:D      2      dnumfiles: 0,      dloadtime: 0,
778:D      2      dlastboot: daterec [year: 0,
779:D      2      day: 0,      month: 0]];
780:S

```

```

781:S
782:D      -4      2      var volsize: integer;
783:D      -8      2      untnumb: integer;
784:D      -12     2      entries: integer;
785:D      -16     2      membytes: integer;
786:D      -16     2      trick: record case integer of
787:D      -16     2      2: (ip: ipointer);
788:D      -16     2      3: (i: integer);
789:D      -20     2      end;
790:D      -682    2      f: fib;
791:D      -762    2      cat: catentry;
792:S
793:C      2      begin { makeramvol }
794:C      2      writeln('*** CREATING A MEMORY VOLUME ***');
795:C      2      writeln('What unit number? ');
796:C      2      if getvalue(7,maxunit,untnumb,true,false) then
797:C      3      begin
798:C      4      write('How many 512 byte BLOCKS? ');
799:C      3      if getvalue(1,maxint,volsize,false,false) then
800:C      4      begin
801:C      5      write('How many entries in directory? ');
802:C      4      if getvalue(0,maxint,entries,false,true) then
803:C      5      begin
804:C      6      releaseuser;
805:C      5      membytes := blocksize*volsize;
806:C      5      if integer(eheap)*membytes > userstack then escape(-2);
807:C      5      newbytes(trick.ip, membytes);
808:C      5      unitable[untnumb] := unitable^[0]; {handles most fields correctly}
809:C      5      with unitable[untnumb] do {fill in the rest}
810:C      6      begin
811:C      7      im := ramdriver;
812:C      6      byteoffset := trick.i;
813:C      6      uvid := 'RAM';
814:C      6      offline := false;
815:C      6      umaxbytes := volsize*fblksize;
816:C      6      end;
817:C      6      with f, cat do
818:C      7      begin
819:C      8      fvid := ''; funit := untnumb; ftitle := '';
820:C      7      fwindow := addr(cat);
821:C      7      cname := 'RAM';
822:C      7      cextral := entries;
823:C      7      cpsize := umaxbytes;
824:C      7      call (dam, f, untnumb, makedirectory);
825:C      7      end;
826:C      6      end;
827:C      3      end;
828:S
829:C      5      if ioresult = ord(inoerror) then
830:C      6      begin
831:C      7      markuser;
832:C      7      writeln;
833:C      6      writeln('#',untnumb:1, ': (RAM:) zeroed');
834:C      6      end
835:C      5      else printerror(-10, ioresult);
836:C      5      end
837:C      3      end;
838:C      2      end; { makeramvol }
839:C
840:S

```



```

841:D 1 procedure newsysunit;
842:D 2 var newunit,i: integer;
843:D -130 2 name: fid;
844:C 2 begin
845:C 2 write('What new system unit number? ');
846:C 2 if getvalue(i, maxunit, newunit, true, false) then
847:C 3 begin
848:C 3 name := '#'; strwrite(name, 2, i, newunit:1, ':');
849:C 3 doprefix(name, syvid, sysunit, true);
850:C 3 if ioresult <> ord(ioerror) then printerror(-10,ioresult)
851:C 4 else updatesysunit;
852:C 3 end;
853:C 2 end; { newsysunit }
854:S
855:D 1 procedure osinit;
856:D -2 2 var iu: 1..maxunit;
857:D -6 2 escocode: integer;
858:C 2 begin {OSINIT}
859:C 2 escocode := sysescapcode;
860:C 2 locklevel := 0; actionspending := 0;
861:C 2 zaptypahead;
862:C 2 if keystream then close(streamfib^);
863:C 3 else close(streamfib^,'PURGE');
864:C 2 for iu := 1 to maxunit do (* force directory cleanups on all vols *)
865:C 3 unitable[iu].umediavalid := false;
866:C 2 sysescapcode := escocode;
867:C 2 end {OSINIT};
868:S
869:D 1 procedure go_prog (debugging: boolean);
870:D -1 2 var stopgoing: boolean;
871:D -6 2 lastioresult: integer;
872:D -8 2 escocode: shortint;
873:D -12 2 userheap: anyptr;
874:D -16 2 modptr: moddescptr;
875:D -17 2 done: boolean;
876:C 2 begin {GO_PROG}
877:C 2 repeat
878:C 3 clearscreen;
879:C 3 writein(output);
880:C 3 modptr := entrypoint;
881:C 3 repeat
882:C 4 done := modptr^.lastmodule;
883:C 4 if modptr^.startaddr <> 0 then
884:C 5 begin
885:C 5 call(debugger,1,entrypoint^.startaddr,ord(debugging));
886:C 5 mark(userheap);
887:C 5 userprogram(modptr^.startaddr,userstack); (** ALL PROGRAMS ARE ENTERED HERE ***)
888:C 5 escocode := escapecode; lastioresult:=ioresult;
889:C 5 release(userheap);
890:C 5 call(debugger,2,0,0);
891:C 5 stopgoing := true;
892:C 5 openfiles;
893:C 5 if escocode <> 0 then
894:C 6 begin
895:C 6 done := true;
896:C 6 disableuserisrs;
897:C 6 osinit; (*shuts off stream files*)
898:C 6 if (escocode <> -1) and (escocode <> -20) then
899:C 7 begin
900:C 7 printerror(escocode,lastioresult);

```

```

901:C 7 prompt('Restart with debugger ? ');
902:C 7 stopgoing := getchar(false) <> 'Y';
903:C 7 end;
904:C 6 debugging := not stopgoing;
905:C 6 end;
906:C 5 end;
907:C 4 modptr := modptr^.link;
908:C 4 until done;
909:C 3 until stopgoing;
910:C 2 end; {GO_PROG}
911:S
912:D 1 procedure loadandgo (var fileto go:fid; permanent, debugging: boolean);
913:D 2 label i;
914:D -146 2 var vol: vid; name: fid; segs: integer; kind: filekind;
915:D -168 2 modp: moddescptr; upcname: tid;
916:C 2 begin
917:C 2 if not permanent then
918:C 3 if scantitle(fileto go,vol,name,segs,kind) then
919:C 4 begin
920:C 4 if strlen(name) <= tidleng then upcname := name else upcname := '';
921:C 4 upc(upcname);
922:C 4 modp := sysdefs;
923:C 4 while modp <> nil do with modp^ do
924:C 5 begin
925:C 5 if startaddr <> 0 then
926:C 6 if (name = progame) or (ucase and (upcname = progame)) then
927:C 7 begin
928:C 7 if entrypoint <> modp then releaseuser;
929:C 8 entrypoint := modp;
930:C 8 go_prog(debugging);
931:C 8 go to i;
932:C 8 end;
933:C 6 modp := link;
934:C 6 end;
935:C 5 end;
936:C 2 load(fileto go, permanent);
937:C 2 if permanent then markuser;
938:C 2 else if entrypoint <> nil then go_prog(debugging);
939:C 2 i: end; {LOADANDGO}
940:S
941:D 1 procedure initheap;
942:D -4 2 var marker: anyptr;
943:C 2 begin (*BASIC FILE AND HEAP SETUP*)
944:C 2 new(userinfo);
945:C 2 new(streamfib);
946:C 2 mark(marker);
947:C 2 if integer(marker) > userstack then escape(-2);
948:C 2 markuser;
949:C 2 end {INITHEAP};
950:S
951:D 1 procedure initunits;
952:D 2 var
953:D -2 2 lunit: unitnum;
954:D -664 2 f: fib;
955:C 2 begin
956:C 2 f: fileid := 0;
957:C 2 for lunit := 1 to maxunit do
958:C 3 with unitable[lunit] do
959:C 4 begin
960:C 4 offline := false;

```



```

961:C      4      umediavalid := false;
962:C      4      f.funit := lunit;
963:C      4      call (tm, addr(f), clearunit, f, 0, 0);
964:C      4      offline := uisblkd and (ioresult<>0);
965:C      4      end;
966:C      2 end; (*INITUNITS*)
967:S
968:S
969:D      1 procedure command;
970:D      2 type prompttype=string[79];
971:D      2 const
972:D      2      prompt1=prompttype
973:D      2 ['Command: Cmplr Edit File Init Libr Run Xcut Ver ?'];
974:D      2      prompt2=prompttype
975:D      2 ['Command: Asm Dbg Memv New Perm Stream User What ?'];
976:S
977:D      2      lprompt1=prompttype
978:D      2 ['Command: Compiler Editor Filer Initialize Librarian Run eXecute Version ?'];
979:D      2      lprompt2=prompttype
980:D      2 ['Command: Assembler Debugger Memvol Newsysvol Permanent Stream User What ?'];
981:S
982:D      -1 2 var      skipping: boolean;
983:D      -6 2      i: integer;
984:D      -10 2      pl: ^prompttype;
985:D      -11 2      plfirst: boolean;
986:S
987:D      2      procedure execute(permanent: boolean; debugging: boolean);
988:D      -122 3      var title: fid;
989:C      3      begin
990:C      3          if permanent then
991:C      4              prompt('Load what code file? ')
992:C      4          else if debugging then
993:C      5              prompt('Debug what file? ')
994:C      5          else prompt('Execute what file? ');
995:C      3          readln(input,title);
996:C      3          fixname(title,codefile);
997:C      3          if strlen(title) > 0 then
998:C      4              begin
999:C      5                  if strlen(title) > (sizeof(fid)-6)
1000:C      5                  then setstrlen(title, sizeof(fid)-6);
1001:C      4                  loadandgo(title,permanent,debugging);
1002:C      4              end;
1003:C      3          end (*EXECUTE*);
1004:S
1005:D      2      procedure compileandedit;
1006:C      3      begin (COMPILEANDEDIT)
1007:C      3          with userinfo do
1008:C      4              begin
1009:C      4                  errnum := 0; errblk := 0;
1010:C      4                  loadandgo(filename[compiler],false,false);
1011:C      4                  if entrypoint <> nil then if errnum <> 0 then
1012:C      6                      loadandgo(filename[editor],false,false);
1013:C      4                  end;
1014:C      3          end; (COMPILEANDEDIT)
1015:S
1016:D      2      procedure runworkfile (debugging:boolean);
1017:D      -122 3      var title: fid;
1018:C      3      begin with userinfo do
1019:C      4          if not (gotsym or gotcode) then execute(false,debugging)
1020:C      5          else

```

```

1021:C      5      begin
1022:C      5          if not gotcode then compileandedit;
1023:C      5          if gotcode then
1024:C      6              begin
1025:C      6                  title:=codefid;
1026:C      6                  loadandgo(title,false,debugging);
1027:C      6              end;
1028:C      5          end;
1029:C      3      end (*RUNWORKFILE*);
1030:S
1031:D      2      procedure stream;
1032:D      3      var
1033:D      -87 3      sfile: string80; i: integer; done: boolean;
1034:C      3      begin
1035:C      3          if chaining = streamchain then
1036:C      4              begin
1037:C      4                  chaining := nochain;
1038:C      4                  sfile := chainfile;
1039:C      4              end
1040:C      4          else
1041:C      4              begin
1042:C      4                  prompt('Stream what file ? ');
1043:C      4                  readln(input,sfile);
1044:C      4                  end;
1045:C      3          if strlen(sfile) <= 70 then          {too long of name can crash system}
1046:C      4              fixname(sfile,txtfile)      { since SFILE is of type STRING80 }
1047:C      4          else
1048:C      4              begin
1049:C      4                  sfile := '';
1050:C      4                  writeln(output,'Stream file name too long');
1051:C      4                  end;
1052:C      3          if strlen (sfile) > 0 then          {GOT VALID NAME}
1053:C      4              begin
1054:C      4                  if keystream then close(streamfib^);
1055:C      5                  else close(streamfib^, 'PURGE');
1056:C      4                  streamopen(sfile,true);      {TRY TO OPEN THE STREAM FILE}
1057:C      4                  end;                          {EXIT IF NO NAME GIVEN}
1058:C      3          end; (*STREAM*)
1059:S
1060:D      2      begin (COMMAND)
1061:C      2          skipping := false;
1062:C      2          plfirst:=true;
1063:C      2          cl_cmd:=' ';
1064:C      2          call(cmdcharhook^);
1065:C      2          repeat
1066:C      3              if chaining=progchain then
1067:C      4                  begin
1068:C      4                      chaining := nochain;
1069:C      4                      loadandgo(chainfile,false,false);
1070:C      4                  end
1071:C      4              else if chaining=streamchain then stream
1072:C      5              else
1073:C      5                  begin
1074:C      6                      if skipping then
1075:C      6                          begin
1076:C      6                              while streaming and skipping do
1077:C      7                                  begin
1078:C      7                                      if eoln(input) then
1079:C      8                                          begin
1080:C      8                                              get(input);

```



```

1081:C      8      if streaming then skipping := input^ = '*';
1082:C      8      end
1083:C      8      else get(input);
1084:C      7      end;
1085:C      6      for i := 1 to 80000 do (nothing);
1086:C      6      skipping := false;
1087:C      6      end;
1088:C      5
1089:C      5      if syscom^.crtinfo.width>=80 then
***WARNING: (line 1090): 'ADDR' of a constant may not be supported on other implementations
***WARNING: (line 1090): 'ADDR' of a constant may not be supported on other implementations
1090:C      3      if plfirst then pl:=addr(lprompt1) else pl:=addr(lprompt2)
1091:C      7      else
***WARNING: (line 1092): 'ADDR' of a constant may not be supported on other implementations
***WARNING: (line 1092): 'ADDR' of a constant may not be supported on other implementations
1092:C      6      if plfirst then pl:=addr(prompt1) else pl:=addr(prompt2);
1093:C      5
1094:C      5      if ci_cmd<>chr(0) then prompt(pl^);
1095:C      5      ci_idle:=true;
1096:C      5      disptime;
1097:C      5      if chaining=nochain then
1098:C      5      begin
1099:C      6      ci_cmd := getchar(false); call(cmdcharhook^);
1100:C      6      if ci_cmd<>chr(0) then
1101:C      6      begin clearscreen; writeln(output); end;
1102:C      6      end;
1103:C      6
1104:C      5
1105:C      5      if ci_cmd = chr(3) then ci_cmd = 'X';
1106:C      5
1107:C      5      setrunlight(ci_cmd);(set the run light to indicate command)
1108:C      5
1109:C      5      if not ((ci_cmd=' ') or (ci_cmd=chr(0))) then
1110:C      6      case ci_cmd of
1111:C      7      '?: plfirst := not plfirst;
1112:C      7      'A': loadandgo(filename[assembly],false,false);
1113:C      7      'C': compileandedit;
1114:C      7      'D': runworkfile(true);
1115:C      7      'E': begin
1116:C      7      userinfo^.errnum := 0;
1117:C      7      userinfo^.errblk := 0;
1118:C      7      loadandgo(filename[editor],false,false);
1119:C      7      end;
1120:C      7      'F': loadandgo(filename[filer],false,false);
1121:C      7
1122:C      7      'I': begin
1123:C      7      lockup;
1124:C      7      releaseuser;
1125:C      7      lockfiles;
1126:C      7      initunits;
1127:C      7      openfiles;
1128:C      7      lockdown;
1129:C      7      end;
1130:C      7      'L': loadandgo(filename[librarian],false,false);
1131:C      7      'M': makeramvol;
1132:C      7      'N': newsysunit;
1133:C      7      'O': execute(true,false);
1134:C      7      'R': runworkfile(false);
1135:C      7      'S': stream;
1136:C      7      'U': if entrypoint <> nil then go_prog(false)

```

```

1137:C      8      else execute(false, false);
1138:C      7      'V': dateset;
1139:C      7      'W': whatfiles;
1140:C      7      'X': execute(false,false);
1141:C      7      otherwise
1142:C      7      if streaming then
1143:C      8      if ci_cmd = '*' then
1144:C      9      begin
1145:C      9      skipping := true;
1146:C      9      write(output,'*');
1147:C      9      end
1148:C      9      else
1149:C      9      begin
1150:C      9      osinit;
1151:C      9      if (ci_cmd > ' ') and (ord(ci_cmd)<127) then
1152:C      10      write(output,' ',ci_cmd,' ');
1153:C      10      else
1154:C      10      write(output,'Character #',ord(ci_cmd));
1155:C      9      write(output,' is not a command. ');
1156:C      9      end;
1157:C      7      end; (CASE5)
1158:C      5      end;
1159:C      3      until false;
1160:C      2      end; (COMMAND)
1161:C      5
1162:C      5
1163:C      5
1164:C      1      procedure systemstartup;
1165:C      2      var done: boolean;
1166:C      2      begin
1167:C      2      initheap; (* point of final allocation of heap space *)
1168:C      2      repeat
1169:C      3      try
1170:C      4      call(debugger,2,0,0); ( log in with debugger )
1171:C      4      chaining := nochain;
1172:C      4      versionup := false;
1173:C      4      ci_idle:=false;
1174:C      4      if cmdcharhook=nil then begin
1175:C      5      new(cmdcharhook); markuser;
1176:C      5      cmdcharhook^ := dummycmdchar;
1177:C      5      end;
1178:C      4      initworkfile;
1179:C      4      initfnames;
1180:C      4      streamopen('AUTOSTART',false); (open autostart stream file before dateset)
1181:C      4      if ioresult<>ord(ioerror) then streamopen('*AUTOKEYS[*]',false);
1182:C      4      initdate;
1183:C      4      dateset;
1184:C      5
1185:C      4      repeat
1186:C      5      try
1187:C      6      command
1188:C      6      recover
1189:C      6      repeat
1190:C      7      try
1191:C      8      ioresult:=ioresult; (save it)
1192:C      8      call(debugger,2,0,0);
1193:C      8      osinit; (shut off stream files)
1194:C      8      if escapecode <> -1 then
1195:C      9      begin
1196:C      9      disableuserisrs;

```



```
1197:C      9      clearscreen; writeln(output);
1198:C      9      printerror(escapecode,ioresult);
1199:C      9      writeln(output,'Trapped by outer level of OS.');
```

```
1200:C      9      end;
1201:C      8      done := true;
1202:C      8      recover done := false;
1203:C      7      until done;
1204:C      5      until false;
1205:C      4      recover printerror(escapecode,ioresult);
1206:C      3      until false;
1207:C      2      end; (*systemstartup*)
1208:S
1209:C      1      end (*MODULE C1*);
1210:S
1211:D      1      import ci,asm;
1212:S
1213:C      1      begin
1214:C      1      ci_switch;
1215:C      1      systemstartup;
1216:C      1      end. (*COMMAND INTERPRETER*)
1217:S
```

No errors. 4 warnings.

***** Nonstandard language features enabled *****

MINIT

Description

MINIT contains the Pascal portion of all the device-dependent drivers in `MEDIAINIT`. Having no real source of its own, this short file includes the files:

- `MIDECS`: Mediainit declarations.
- `MMINIT`: Internal minifloppy initialization drivers.
- `HMINIT`: Supported Amigo device initialization drivers.
- `QMINIT`: CS80 initialization drivers.
- `XMINIT`: Unsupported Amigo device initialization drivers.

Notes

Includes `MMINIT`, `HMINIT`, `QMINIT`, `XMINIT` and `MIDECS`.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:0 0 $modcal$
2:0 0 $debug off, range off, ovflcheck off$
3:0 0 $stackcheck off, iocheck off$
4:0
5:0 0 $search 'IOLIB:KERNEL', 'INIT:DRVASH', 'INIT:DISCHPIB',
6:0 'INIT:AMIGO', 'INIT:CS80'$
7:0
8:0 0 $include 'MIDECS'$;
9:0
10:0 0 module midecs; {mediainit declarations}
11:0
12:0 1 import
13:0 1 sysglobals;
14:0
15:0 1 export
16:0
17:0 1 type
18:0
19:0 1   interleave_data =
20:0 1   record
21:0 1     min: 0..255;   {minimum interleave factor}
22:0 1     max: 0..255;   {maximum interleave factor}
23:0 1     def: 0..255;   {default interleave factor}
24:0 1   end;
25:0
26:0 1   physical_data = {(for LIF volume labels)}
27:0 1   record
28:0 1     ntps: integer; {number of tracks per surface}
29:0 1     nspm: integer; {number of surfaces per media}
30:0 1     nspt: integer; {number of sectors per track}
31:0 1   end;
32:0
33:0
34:0 1   var
35:0 -1 1   extended_features_mode: boolean;
36:0
37:0 -1 1   function yes(prompt: string255): boolean;
38:0
39:0 -1 1   implement {midecs}
40:0
41:0 -256 1   function yes(prompt: string255): boolean;
42:0 -256 2   var
43:0 -257 2     answer: char;
44:0 2     begin {yes}
45:0 2       write(prompt, ' (Y/N) ');
46:0 2       repeat
47:0 2         read(keyboard, answer);
48:0 2         until answer in ['Y', 'y', 'N', 'n'];
49:0 2       writeln(answer);
50:0 2       yes := (answer='Y') or (answer='y');
51:0 2     end; {yes}
52:0
53:0 1   end {midecs}
54:0
55:0 2 $include 'MIDECS'$;
56:0 1 $include 'BMINIT'$;
57:0 1 MODULE BMINIT;
58:0 1 IMPORT SYSGLOBALS, MIDECS;
59:0 1 EXPORT

```

```

59:0 1   PROCEDURE BMINITIALIZE(UN:INTEGER);
60:0 1   FUNCTION BPHYDATA(UN:INTEGER):PHYSICAL_DATA;
61:0 1   IMPLEMENT
62:0 1   PROCEDURE BMINITIALIZE(UN:INTEGER);
63:0 2   VAR
64:0 -4 2     VFILE : FIBP;
65:0 -8 2     HP : ANYPTR;
66:0 -12 2     I : INTEGER;
67:0 -268 2     SECTOR : PACKED ARRAY[0..255] OF CHAR;
68:0 2   BEGIN
69:0 2     MARK(HP);
70:0 2     TRY
71:0 3       NEW(VFILE);
72:0 3       WITH VFILE, UNITABLE^[UN] DO
73:0 4         BEGIN
74:0 4           { INITIALIZE FIB FIELDS }
75:0 4           FUNIT := UN;
76:0 4           FPEOF := UMAYBYTES;
77:0 4           FILEID := 0;
78:0 4           { RESET THE UNIT }
79:0 4           CALL(TM,VFILE,CLEARUNIT,SECTOR,0,0);
80:0 4           IF IORESULT<>ORD(INOERROR) THEN ESCAPE(-10);
81:0
82:0 4           { WRITE BLANKS ALL OVER THE DEVICE }
83:0 4           FOR I := 0 TO 255 DO SECTOR[I] := ' '; { BLANK FILL }
84:0 4           FOR I := 0 TO (UMAYBYTES DIV 256)-1 DO
85:0 5             BEGIN
86:0 5               CALL(TM,VFILE,WRITEBYTES,SECTOR,256,I*256);
87:0 5               IF IORESULT<>ORD(INOERROR) THEN ESCAPE(-10);
88:0 5             END;
89:0 4           END;
90:0 3       RELEASE(HP);
91:0 3       RECOVER
92:0 3       BEGIN RELEASE(HP); ESCAPE(ESCAPECODE); END;
93:0 2   END;
94:0
95:0 1   FUNCTION BPHYDATA(UN:INTEGER):PHYSICAL_DATA;
96:0 2   VAR
97:0 -12 2     PDATA : PHYSICAL_DATA;
98:0 2   BEGIN
99:0 2     PDATA.NTPS := 1; PDATA.NSPM := 1;
100:0 2     PDATA.NSPT := UNITABLE^[UN].UMAYBYTES DIV 256;
101:0 2     BPHYDATA := PDATA;
102:0 2   END;
103:0
104:0 1   END
105:0 2 $include 'BMINIT'$;
106:0 1 $include 'MINIT'$;

```



```

106:D      1 $pages
107:S
108:D      1 module Minit;
109:S
110:D      1 import
111:D      1   sysglobals, fs, asm, sysdevs, mini, midecs;
112:S
113:D      1 export
114:D      1   function Mintdata: interleave_data;
115:D      1   function Mphysdata: physical_data;
116:D      1   procedure Minitialize(interleave: shortint);
117:S
118:D      1 implement {Minit}
119:S
120:S
121:D      1 function Mintdata: interleave_data;
122:D      2 const
123:D      2   intdata = interleave_data
124:D      2   [ min: 1;           {minimum interleave factor}
125:D      2     max: 15;          {maximum interleave factor}
126:D      2     def: 1 ];         {default interleave factor}
127:C      2 begin {Mintdata}
128:C      2   Mintdata := intdata;
129:C      2 end; {Mintdata}
130:S
131:S
132:D      1 function Mphysdata: physical_data;
133:D      2 const
134:D      2   physdata = physical_data
135:D      2   [ ntps: (2*35-4) div 2, {number of tracks per surface}
136:D      2     nsfm: 2;              {number of surfaces per media}
137:D      2     nspt: 16 ];          {number of sectors per track}
138:C      2 begin {Mphysdata}
139:C      2   Mphysdata := physdata;
140:C      2 end; {Mphysdata}

```

```

141:D      1 $pages
142:S
143:D      1 procedure Minitialize(interleave: shortint);
144:S
145:D      2 { NOTE: the BOOTROM global that selects drive 0 or 1 must already be setup! }
146:S
147:D      2 var
148:D      2   x,y: integer;
149:D      2   crtptr: anyptr;
150:D      2   saved_escapecode: shortint;
151:S
152:C      2 begin {Minitialize}
153:S
154:C      2   call(maskopshook,fhimask,0); {enable kbd fasthandshake}
155:S
156:C      2   writeln;
157:C      2   fgetxy(output,x,y);
158:C      2   with syscom^.crtinfo do
159:C      3     crtptr := anyptr(crtmemaddr+(y-1)*2*width);
160:S
161:C      2   try
162:C      3     flpyinit(crtptr,interleave);
163:C      3     call(maskopshook,0,fhimask); {disable kbd fasthandshake}
164:S
165:C      3   recover
166:C      3   begin
167:C      3     call(maskopshook,0,fhimask); {disable kbd fasthandshake}
168:C      3     if iorval<>inoerror then ioresc(iorval);
169:C      3     saved_escapecode := escapecode;
170:C      3     try
171:C      4       pwr_on;
172:C      4     recover
173:C      4     {ignore any errors};
174:C      3     escape(saved_escapecode);
175:C      3   end; {recover}
176:S
177:C      2 end; {Minitialize}
178:S
179:C      1 end {Minit}
180:S
181:C      2 $include 'MMINIT'$;
181:D      1 $include 'MMINIT'$;

```



```

182:D      1 $page$
183:S
184:D      1 module IRamigo; {amigo initialization routines}
185:S
186:D      1 import
187:D      1   sysglobals, bkgnd, dischPIB, CSamigo;
188:S
189:D      1 export
190:D      1   procedure verify      (uep: uep_type; nsectors: unsigned16);
191:D      1   procedure init_with_d_bits(uep: uep_type);
192:D      1   procedure format      (uep: uep_type; intlve, d_byte: byte; override: boolean);
193:S
194:D      1 implement {IRamigo}
195:S
196:D      1 procedure verify(uep: uep_type; nsectors: unsigned16);
197:D      2   var
198:D      2     verify_cmd_buf:
199:D      2       packed record
200:D      2         ftcb: ftcb_type;
201:D      2         sector_count: unsigned16;
202:D      2       end;
203:D      -4   begin {verify}
204:D      2     verify_cmd_buf.sector_count := nsectors;
205:D      2     issue_cmd(uep, verify_cmd, verify_cmd_buf.ftcb);
206:D      2   end; {verify}
207:S
208:D      1 procedure init_with_d_bits(uep: uep_type);
209:D      2   var
210:D      2     init_cmd_buf: ftcb_type;
211:D      -2     dummy_byte: packed record b: byte; end;
212:D      -4   const
213:D      -4     receive_data_sec := 0;
214:D      2   begin {init_with_d_bits}
215:D      2     issue_cmd(uep, init_d_cmd, init_cmd_buf);
216:D      2     HPI8wait_for_ppol(uep);
217:D      2     dummy_byte.b := 0;
218:D      2     HPI8short_msg_out(uep, receive_data_sec, addr(dummy_byte), sizeof(dummy_byte));
219:D      2   end; {init_with_d_bits}
220:S
221:D      1 procedure format(uep: uep_type; intlve, d_byte: byte; override: boolean);
222:D      2   var
223:D      2     format_cmd_buf:
224:D      2       packed record
225:D      2         ftcb: ftcb_type;
226:D      2         f: boolean; { override old format bit }
227:D      2         tttttt: 0..127; { wanted format type }
228:D      2         int: byte; { interleave factor }
229:D      2         db: byte; { data byte }
230:D      -6     end;
231:D      2   begin {format}
232:D      2     with format_cmd_buf do
233:D      2       begin
234:D      2         f := override;
235:D      2         tttttt := 2; { HP format always }
236:D      2         int := intlve;
237:D      2         db := d_byte;
238:D      2       end; {with}
239:D      2     issue_cmd(uep, format_cmd, format_cmd_buf.ftcb);
240:D      2   end; {format}
241:S

```

```

242:C      1 end; {IRamigo}
243:S
244:S
245:D      1 module Hminit;
246:S
247:D      1 import
248:D      1   sysglobals, bkgnd, dischPI8, CSamigo, IRamigo, midecs;
249:S
250:D      1 export
251:D      1   function Hintdata: interleave_data;
252:D      1   function Nintdata(uep: uep_type): interleave_data;
253:D      1   function Hphydata(uep: uep_type): physical_data;
254:D      1   procedure Hminitialize(uep: uep_type; intlve: shortint);
255:S
256:D      1 implement {Hminit}
257:S
258:D      1 function Hintdata: interleave_data;
259:D      2   const
260:D      2     intdata := interleave_data
261:D      2       [ min: 1, {minimum interleave factor}
262:D      2         max: 28, {maximum interleave factor}
263:D      2         def: 2 ]; {default interleave factor}
264:D      2   begin {Hintdata}
265:D      2     Hintdata := intdata;
266:D      2   end; {Hintdata}
267:C
268:S
269:S
270:D      1 function Nintdata(uep: uep_type): interleave_data;
271:D      2   const
272:D      2     intdata := interleave_data
273:D      2       [ min: 1, {minimum interleave factor}
274:D      2         max: 16, {maximum interleave factor}
275:D      2         def: 3 ]; {default interleave factor}
276:D      2   begin {Nintdata}
277:D      2     Nintdata := intdata;
278:D      2     if uep.devid > 0 then {this is actually a Sparrow...}
279:D      2       Nintdata.def := 2; {which performs best at this interleave}
280:D      2   end; {Nintdata}
281:S
282:S
283:D      1 function Hphydata(uep: uep_type): physical_data;
284:D      2   var
285:D      -6     map: map_type;
286:D      -18     phydata: physical_data;
287:D      2   begin {Hphydata}
288:D      2     get_map(uep, map);
289:D      2     with map, phydata do
290:D      2       begin
291:D      2         nspm := trk_per_cyl; {number of surfaces per medium}
292:D      2         ntps := cyl_per_med; {number of tracks/surface}
293:D      2         nsp1 := sec_per_trk; {number of sectors per track}
294:D      2       end; {with}
295:D      2     Hphydata := phydata;
296:C      2   end; {Hphydata}

```



```

297:D      1 $page$
298:S
299:D      1 procedure Hminitialize(uep: uep_type; intlv: shortint);
300:S
301:D      2
302:D      2   var
303:D      2     status_bytes: status_type;
304:D      2     map: map_type;
305:D      2     tracks_per_medium: shortint;
306:D      2
307:C      2   procedure confirm_preeoi;
308:C      2   begin (confirm_preeoi)
309:C      2     with btp_type(uep^.dvrtemp)^ do
310:C      2     if (escapcode=-10) and (iores=zbadhardware)
311:C      2     then iores := ioerror
312:C      2     else escape(escapcode);
313:D      2   end; (confirm_preeoi)
314:D      2
315:D      2   procedure check_status;
316:C      2   begin (check_status)
317:C      2     status(uep, status_bytes);
318:C      2     case status_bytes.51 of
319:C      2     ( 'acceptable' errors)
320:C      2     ( 'uncorrectable_data_error',
321:C      2     ( 'head_sector_compare_error',
322:C      2     ( 'access_not_ready_during_data_operation:
323:C      2     ( 'acceptable; do not escape);
324:C      2     (error requiring status 2 check)
325:C      2     drive_attention;
326:C      2     if not status_bytes.c then ioresc_bkgnd(uep, zinitfail);
327:C      2     (other errors)
328:C      2     otherwise ioresc_bkgnd(uep, zinitfail)
329:C      2     end; (case)
330:S      2   end; (check_status)
331:S
332:D      2
333:D      2   procedure init_9895_913X_Chinook;
334:D      2   const
335:D      2     testpasses = 4;           {number of test passes}
336:D      2     max_bt = 4;             {maximum number of bad tracks}
337:D      2   type
338:D      2     test_pattern_type = packed array[1..testpasses] of byte;
339:D      2   const
340:D      2     test_pattern = test_pattern_type
341:D      2     [ 198,   { $C6 or 1100011011000110 }
342:D      2       99,    { $63 or 0110001101100011 }
343:D      2       219,   { $DB or 1101101111011011 }
344:D      2       136 ]; { $88 or 1000100010001000 }
345:D      2   var
346:D      2     bt_count: shortint;
347:D      2     test_pass: shortint;
348:D      2     verify_starting_track: shortint;
349:D      2     verify_pass_complete: boolean;
350:D      2     bt_already_logged: boolean;
351:D      2     bad_track: shortint;
352:D      2     bt_index: shortint;
353:D      2     bt_table: array[1..max_bt] of shortint;

```

```

353:D      -20 3 $page$
354:S
355:C      3   begin (init_9895_913X_Chinook)
356:C      3     bt_count := 0;
357:C      3     for test_pass := 1 to testpasses do
358:C      3     begin
359:C      3       format(uep, 1, test_pattern[test_pass], true);
360:C      3       HPIBwait_for_ppol(uep);
361:C      3       if dsj(uep)<>0 then ioresc_bkgnd(uep, zinitfail);
362:C      3       verify_starting_track := 0;
363:C      3       repeat
364:C      3       seek(uep, verify_starting_track*map.sec_per_trk);
365:C      3       HPIBwait_for_ppol(uep);
366:C      3       verify(uep, (tracks_per_medium*max_bt-verify_starting_track)*map.sec_per_trk);
367:C      3       HPIBwait_for_ppol(uep);
368:C      3       if dsj(uep)=0 then
369:C      3       verify_pass_complete := true
370:C      3       else
371:C      3       begin
372:C      3         check_status;
373:C      3         bad_track := logical_addr(uep) div map.sec_per_trk;
374:C      3         bt_already_logged := false; {initial assumption}
375:C      3         for bt_index := 1 to bt_count do
376:C      3         if bt_table[bt_index]=bad_track then bt_already_logged := true;
377:C      3         if not bt_already_logged then (log it)
378:C      3         begin
379:C      3           if bt_count>max_bt then ioresc_bkgnd(uep, zinitfail);
380:C      3           bt_count := bt_count+1;
381:C      3           bt_table[bt_count] := bad_track;
382:C      3         end;
383:C      3         verify_starting_track := bad_track+1;
384:C      3         verify_pass_complete := verify_starting_track>tracks_per_medium*max_bt;
385:C      3       end (else)
386:C      3     until verify_pass_complete;
387:C      3   end; (for)
388:S
389:C      3   for bt_index := 1 to bt_count do {set all d-bits on the bad tracks}
390:C      3   begin
391:C      3     seek(uep, bt_table[bt_index]*map.sec_per_trk);
392:C      3     HPIBwait_for_ppol(uep);
393:C      3     init_with_d_Bits(uep);
394:C      3     HPIBwait_for_ppol(uep);
395:C      3     if dsj(uep)<>0 then ioresc_bkgnd(uep, zinitfail);
396:C      3   end;
397:S
398:C      3   format(uep, intlv, 0, false);
399:C      3   HPIBwait_for_ppol(uep);
400:C      3   if dsj(uep)<>0 then ioresc_bkgnd(uep, zinitfail);
401:C      3   seek(uep, 0);
402:C      3   HPIBwait_for_ppol(uep);
403:C      3   verify(uep, tracks_per_medium*map.sec_per_trk);
404:C      3   HPIBwait_for_ppol(uep);
405:C      3   if dsj(uep)<>0 then ioresc_bkgnd(uep, zinitfail);
406:C      3   end; (init_9895_913X_Chinook)

```



```

407:D -12 2 $page$
408:S
409:D 2 procedure init_913X_ABC;
410:S
411:D 3
412:D -2 3 var
413:D -4 3 verify_pass: shortint;
414:D -6 3 csvp: shortint; (consecutive successful verify passes)
415:D -7 3 verify_starting_track: shortint;
416:S
417:D -7 3 const
418:D -7 3 min_vp = 5; (minimum # of verify passes)
419:D -7 3 max_vp = 15; (maximum # of verify passes)
420:D -7 3 min_csvp = 2; (minimum # of consecutive successful verify passes)
421:S
422:C 3 begin {init_913X_ABC}
423:S
424:C 3 format(uep, intive, 0, true);
425:C 3 HPIBwait_for_ppol(uep);
426:C 3 if dsj(uep)<>0 then ioresc_bkgnd(uep, zinitfail);
427:S
428:C 3 verify_pass := 0;
429:C 3 csvp := 0;
430:S
431:C 3 repeat
432:S
433:C 4 verify_pass := verify_pass+1;
434:C 4 verify_starting_track := 0;
435:S
436:C 4 repeat
437:C 5 seek(uep, verify_starting_track*map.sec_per_trk);
438:C 5 HPIBwait_for_ppol(uep);
439:C 5 verify(uep, (tracks_per_medium-verify_starting_track)*map.sec_per_trk);
440:C 5 HPIBwait_for_ppol(uep);
441:C 5 if dsj(uep)=0 then
442:C 6 verify_pass_complete := true
443:C 6 else
444:C 6 begin
445:C 6 check_status;
446:C 6 verify_starting_track := logical_addr(uep) div map.sec_per_trk+1;
447:C 6 init_with_d_bits(uep);
448:C 6 HPIBwait_for_ppol(uep);
449:C 6 if dsj(uep)<>0 then ioresc_bkgnd(uep, zinitfail);
450:C 6 verify_pass_complete := verify_starting_track=tracks_per_medium;
451:C 6 end {else}
452:C 6 until verify_pass_complete;
453:S
454:C 4 if verify_starting_track=0 {i.e., successful verify pass}
455:C 5 then csvp := csvp+1
456:C 5 else csvp := 0;
457:S
458:C 4 until ((verify_pass>=min_vp) and (csvp>=min_csvp)) or (verify_pass>=max_vp);
459:S
460:C 3 if csvp<min_csvp then ioresc_bkgnd(uep, zinitfail);
461:S
462:C 3 end; {init_913X_ABC}

```

```

463:D -12 2 $page$
464:S
465:C 2 begin {Hminitialize}
466:C 2 try
467:C 3 allocate_bkgnd_info(uep);
468:S
469:C 3 try
470:C 4 status(uep, status_bytes);
471:C 4 recover
472:C 4 begin
473:C 4 confirm_preedi;
474:C 4 if dsj(uep)=2 then ioresc_bkgnd(uep, zmediumchanged);
475:C 5 else ioresc_bkgnd(uep, zbadhardware);
476:C 4 end; {recover}
477:S
478:C 3 with status_bytes do
479:C 4 begin
480:C 4 if f then ioresc_bkgnd(uep, zmediumchanged);
481:C 4 if ss<>0 then ioresc_bkgnd(uep, znomedium);
482:C 4 if w then ioresc_bkgnd(uep, zprotected);
483:C 4 end; {with}
484:S
485:C 3 get_map(uep, map);
486:C 3 with map do
487:C 4 tracks_per_medium := trk_per_cyl*cyl_per_med;
488:S
489:C 3 case device(uep) of
490:C 4 HP9895, HP8290X: init_9895_913X_Chinook;
491:C 4 HP913X_A..HP913X_C: init_913X_ABC;
492:C 4 otherwise ioresc_bkgnd(uep, zcatchall);
493:C 4 end; {case}
494:S
495:C 3 deallocate_bkgnd_info(uep);
496:S
497:C 3 recover
498:C 3 begin
499:C 3 abort_bkgnd_process(uep);
500:C 3 ioresult := uep^dvrtemp;
501:C 3 uep^dvrtemp := ord(inoerror); (report the error only once)
502:C 3 escape(-10);
503:C 3 end; {recover}
504:S
505:C 2 end; {Hminitialize}
506:S
507:C 1 end {Hminit}
508:S
509:S
510:C 2 $include 'HMINIT'$;
510:D 1 $include 'QMINIT'$;

```



```

S11:D      1 $page$
S12:S
S13:D      1 module CS80ir; (Command Set '80 initialization routines)
S14:S
S15:D      1 import
S16:D      1 sysglobals, bkgnd, discHPIB, CS80;
S17:S
S18:D      1 export
S19:D      1 type
S20:D      1   tva_type = (three-vector address (6 bytes))
S21:D      1   packed record
S22:D      1     cylin: unsn24;
S23:D      1     head: unsn8;
S24:D      1     sect: signed16;
S25:D      1   end;
S26:D
S27:D      1   de_log_type = (the two distinct types of data error logs)
S28:D      1   (ert, runtime);
S29:D
S30:D      1   log_entry_type = (data error log entry)
S31:D      1   packed record
S32:D      1     physical_cylin: signed16;
S33:D      1     physical_head: unsn8;
S34:D      1     physical_sect: unsn8;
S35:D      1     logical_cylin: signed16;
S36:D      1     logical_head: unsn8;
S37:D      1     logical_sect: unsn8;
S38:D      1     error_byte: unsn8;
S39:D      1     occurrences: unsn8;
S40:D      1   end;
S41:D
S42:D      1   log_info_type = (info returned by the READ_DATA_ERROR_LOGS command)
S43:D      1   packed record
S44:D      1     my_special_pad: unsn8; (internal alignment only; not sent by the device!!!)
S45:D      1     log_entries: unsn8;
S46:D      1     other_info: array[1..5*2+1] of char;
S47:D      1     entry: array[1..106] of log_entry_type;
S48:D      1   end;
S49:D
S50:D      1   ert_area_type =
S51:D      1   (sector_ert, track_ert, cylinder_ert, surface_ert, volume_ert);
S52:D
S53:S
S54:S      { NOTE: the following functions each perform a COMPLETE transaction. They:
S55:S      . issue a (device or transparent) command (Command message)
S56:S      . transfer data if applicable (Execution message)
S57:S      . return the resulting QSTAT (Reporting message)
S58:D      1 }
S59:D      1 function formatting_option (uep: uep_type; option: unsn8): unsn8;
S60:D      1 function initiate_diagnostic(uep: uep_type; unit: unsn4; loops: signed16; section: unsn8): unsn8;
S61:D      1 function initialize_media (uep: uep_type; options, interleave: unsn8): unsn8;
S62:D      1 function preset_drive (uep: uep_type): unsn8;
S63:D      1 function clear_logs (uep: uep_type; logcode: unsn8): unsn8;
S64:D      1 function set_3v_address (uep: uep_type; cylin: unsn24; head: unsn8; sect: signed16): unsn8;
S65:D      1 function data_error_log (uep: uep_type; de_log: de_log_type; head: unsn8;
S66:D      2   var log_bytes: log_info_type): unsn8;
S67:D      2   (uep: uep_type; ert_area: ert_area_type; loops: unsn8;
S68:D      2   var ert_message: boolean; var log_entry: log_entry_type): unsn8;
S69:D      1 function spare_block (uep: uep_type; sparingmode: unsn8): unsn8;

```

```

S70:D      1 $page$
S71:S
S72:D      1 implement (CS80ir)
S73:S
S74:D      1 type
S75:D      1   setunitvol_type = (SET_UNIT/SET_VOLUME command pair)
S76:D      1   packed record
S77:D      1     setunit: CMD_type;
S78:D      1     setvol: CMD_type;
S79:D      1   end;
S80:D
S81:S
S82:D      1 function suv_CMD_pair(uep: uep_type): setunitvol_type;
S83:D      2 begin {suv_CMD_pair}
S84:C      2   suv_CMD_pair.setunit := CMD_type(signed16(CMDset_unit_0)+uep^.du);
S85:C      2   suv_CMD_pair.setvol := CMD_type(signed16(CMDset_vol_0)+uep^.dv);
S86:C      2 end; {suv_CMD_pair}
S87:C
S88:S
S89:D      1 function formatting_option(uep: uep_type; option: unsn8): unsn8;
S90:D
S91:D      { issue the following command sequence:
S92:D      . SET_UNIT
S93:D      . SET_VOLUME
S94:D      . SET_FORMAT_OPTION (Subset/80 only)
S95:D      issue the option byte in an execution message
S96:D      return the QSTAT byte
S97:D      }
S98:D      2 }
S99:D      2 var
S100:D      2   fo: (the 5 bytes in the command message)
S101:D      2   packed record
S102:D      2     setunitvol: setunitvol_type;
S103:D      2     initutil: CMD_type;
S104:D      2     setformatoption: unsn8;
S105:D      2     parameter: unsn8;
S106:D      2   end;
S107:D      2   em: (the single byte execution message)
S108:D      2   packed record
S109:D      2     optionbyte: unsn8;
S110:D      2   end;
S111:D      2 begin {formatting_option}
S112:C      2   fo.setunitvol := suv_CMD_pair(uep);
S113:C      2   fo.initutil := CMDinit_util_REH;
S114:C      2   fo.setformatoption := 243;
S115:C      2   fo.parameter := 95;
S116:C      2   HPIShort_msge_out(uep, command_sec, addr(fo), sizeof(fo));
S117:C      2   HPISWait_for_ppol(uep);
S118:C      2   em.optionbyte := option;
S119:C      2   HPISShort_msge_out(uep, execution_sec, addr(em), sizeof(em));
S120:C      2   HPISWait_for_ppol(uep);
S121:C      2   formatting_option := qstat(uep);
S122:C      2 end; {formatting_option}

```



```

623:D      1 $page$
624:S
625:D      1 function initiate_diagnostic(uep: uep_type; unit: unsn8; loops: signed16; section: unsn8): unsn8;
626:S      {
627:S          set specified unit & issue the INITIATE_DIAGNOSTIC command
628:S          return the QSTAT byte
629:S      }
630:D      2
631:D      2 var
632:D      2     id: (the 5 bytes in the initiate diagnostic command message)
633:D      2     packed record
634:D      2         setunit: CMD_type;
635:D      2         initdiag: CMD_type;
636:D      2         loops: signed16;
637:D      2         section: unsn8;
638:D      2     end;
639:C      -6
640:C      2 begin (initiate_diagnostic)
641:C      2     id.setunit := CMD_type(signed16(CMDset_unit_0)+unit);
642:C      2     id.initdiag := CMDInit_diagnostic;
643:C      2     id.loops := loops;
644:C      2     id.section := section;
645:C      2     HPIBshort_msge_out(uep, command_sec, addr(id), sizeof(id));
646:C      2     HPIBwait_for_ppol(uep);
647:C      2     initiate_diagnostic := qstat(uep);
648:C      2 end; (initiate_diagnostic)
649:S
650:D      1 function initialize_media(uep: uep_type; options, interleave: unsn8): unsn8;
651:S      {
652:S          issue the following command sequence:
653:S              . SET UNIT
654:S              . SET VOLUME
655:S              . INITIALIZE MEDIA
656:S          return the QSTAT byte
657:S      }
658:D      2
659:D      2 var
660:D      2     im: (the 5 bytes in the command message)
661:D      2     packed record
662:D      2         setunitvol: setunitvol_type;
663:D      2         initmedia: CMD_type;
664:D      2         options: unsn8;
665:D      2         interleave: unsn8;
666:D      2     end;
667:C      -6
668:C      2 begin (initialize_media)
669:C      2     im.setunitvol := suv_CMD_pair(uep);
670:C      2     im.initmedia := CMDInit_media;
671:C      2     im.options := options;
672:C      2     im.interleave := interleave;
673:C      2     HPIBshort_msge_out(uep, command_sec, addr(im), sizeof(im));
674:C      2     HPIBwait_for_ppol(uep);
675:C      2     initialize_media := qstat(uep);
676:C      2 end; (initialize_media)

```

```

674:D      1 $page$
675:S
676:D      1 function preset_drive(uep: uep_type): unsn8;
677:S      {
678:S          issue the following command sequence:
679:S              . SET UNIT
680:S              . SET VOLUME
681:S              . PRESET DRIVE
682:S          return the QSTAT byte
683:S      }
684:D      2
685:D      2 var
686:D      2     pd: (the 4 bytes in the command message)
687:D      2     packed record
688:D      2         setunitvol: setunitvol_type;
689:D      2         initutil: CMD_type;
690:D      2         presetdrive: unsn8;
691:D      2     end;
692:C      -4
693:C      2 begin (preset_drive)
694:C      2     pd.setunitvol := suv_CMD_pair(uep);
695:C      2     pd.initutil := CMDInit_util_NEM;
696:C      2     pd.presetdrive := 206;
697:C      2     HPIBshort_msge_out(uep, command_sec, addr(pd), sizeof(pd));
698:C      2     HPIBwait_for_ppol(uep);
699:C      2     preset_drive := qstat(uep);
700:C      2 end; (preset_drive)
701:S
702:D      1 function clear_logs(uep: uep_type; logcode: unsn8): unsn8;
703:S      {
704:S          issue the following command sequence:
705:S              . SET UNIT
706:S              . SET VOLUME
707:S              . CLEAR LOGS
708:S          return the QSTAT byte
709:S      }
710:D      2
711:D      2 var
712:D      2     cl: (the 5 bytes in the command message)
713:D      2     packed record
714:D      2         setunitvol: setunitvol_type;
715:D      2         initutil: CMD_type;
716:D      2         clearlogs: unsn8;
717:D      2         logcode: unsn8;
718:D      2     end;
719:C      -6
720:C      2 begin (clear_logs)
721:C      2     cl.setunitvol := suv_CMD_pair(uep);
722:C      2     cl.initutil := CMDInit_util_NEM;
723:C      2     cl.clearlogs := 205;
724:C      2     cl.logcode := logcode;
725:C      2     HPIBshort_msge_out(uep, command_sec, addr(cl), sizeof(cl));
726:C      2     HPIBwait_for_ppol(uep);
727:C      2     clear_logs := qstat(uep);
728:C      2 end; (clear_logs)

```



```

726:D      1 $pages$
727:S
728:D      1 function data_error_log(uep: uep_type; de_log: de_log_type; head: unsgn8;
729:D      2      var log_bytes: log_info_type): unsgn8;
730:S      {
731:S      { issue the appropriate READ_DATA_ERROR_LOG command
732:S      { place the returned data error log info in the passed variable 'log_bytes'
733:S      { return the QSTAT byte
734:D      2      }
735:D      2      type
736:D      2      read_de_log_micro_op_type =
737:D      2      array[de_log_type] of unsgn8;
738:D      2      const
739:D      2      read_de_log_micro_op = read_de_log_micro_op_type
740:D      2      [ (ert) 198, (runtime) 197 ];
741:D      2      var
742:D      2      del: (the 5 bytes in the command message)
743:D      2      packed record
744:D      2      setunitvol: setunitvol_type;
745:D      2      initutil: CMD_type;
746:D      2      readlogs: unsgn8;
747:D      2      head: unsgn8;
748:D      2      end;
749:C      -6      begin (data_error_log)
750:C      2      del.setunitvol := sub_CMD_pair(uep);
751:C      2      del.initutil := CMDinit_util_SEM;
752:C      2      del.readlogs := read_de_log_micro_op[de_log];
753:C      2      del.head := head;
754:C      2      HPIBshort_msge_out(uep, command_sec, addr(del), sizeof(del));
755:C      2      HPIBwait_for_ppol(uep);
756:C      2      try
757:C      3      HPIBshort_msge_in(uep, execution_sec, addr(log_bytes, 1), sizeof(log_bytes)-1);
758:C      3      recover
759:C      4      with bip_type(uep^.dvrtemp)^ do (confirm the "premature" eoi)
760:C      4      begin
761:C      5      if (escapecode<>-10) or (iores<>zbadhardware) then
762:C      5      escape(escapecode);
763:C      4      iores := inoerror;
764:C      4      end; (recover)
765:C      2      HPIBwait_for_ppol(uep);
766:C      2      data_error_log := qstat(uep);
767:C      2      end; (data_error_log)

```

```

768:D      1 $pages$
769:S
770:D      1 function pattern_ert(uep: uep_type; ert_area: ert_area_type; loops: unsgn8;
771:D      2      var ert_message: boolean; var log_entry: log_entry_type): unsgn8;
772:S      {
773:S      { issue the appropriate ERROR_RATE_TEST command
774:S      { if specified request an execution message and set var boolean accordingly
775:S      { return the QSTAT byte
776:D      2      }
777:D      2      var
778:D      2      ert: (the 9 bytes in the command message)
779:D      2      packed record
780:D      2      setunitvol: setunitvol_type;
781:D      2      initutil: CMD_type;
782:D      2      patternert: unsgn8;
783:D      2      loops: unsgn8;
784:D      2      offset: signed8;
785:D      2      report: unsgn8;
786:D      2      testarea: unsgn8;
787:D      2      datasource: unsgn8;
788:D      2      end;
789:C      -10      begin (pattern_ert)
790:C      2      ert.setunitvol := sub_CMD_pair(uep);
791:C      2      if ert_message
792:C      3      then ert.initutil := CMDinit_util_SEM; {send execution message}
793:C      3      else ert.initutil := CMDinit_util_NEM; {no execution message}
794:C      2      ert.patternert := 200; {pattern ert micro opcode}
795:C      2      ert.loops := loops; {as specified}
796:C      2      ert.offset := 0; {no offset}
797:C      2      ert.report := 0; {data error log info only}
798:C      2      ert.testarea := ord(ert_area); {as specified}
799:C      2      ert.datasource := 0; {internal pattern table}
800:C      2      HPIBshort_msge_out(uep, command_sec, addr(ert), sizeof(ert));
801:C      2      if ert_message then
802:C      3      try
803:C      4      HPIBwait_for_ppol(uep);
804:C      4      HPIBshort_msge_in(uep, execution_sec, addr(log_entry), sizeof(log_entry));
805:C      4      recover
806:C      4      with bip_type(uep^.dvrtemp)^ do (confirm the "premature" eoi)
807:C      4      begin
808:C      5      if (escapecode<>-10) or (iores<>zbadhardware) then
809:C      5      escape(escapecode);
810:C      4      iores := inoerror;
811:C      4      ert_message := false;
812:C      4      end; (recover)
813:C      2      HPIBwait_for_ppol(uep);
814:C      2      pattern_ert := qstat(uep);
815:C      2      end; (pattern_ert)

```



```

816:D      1 $page$
817:S
818:D      1 function set_3V_address(uep: uep_type; cylin: unsgn24; head: unsgn8; sect: signed16): unsgn8;
819:S      (
820:S          issue the following command sequence:
821:S              SET_UNIT
822:S              SET_VOLUME
823:S              SET_ADDRESS
824:S          return the QSTAT byte
825:S      )
826:D      var
827:D          s3va: (the 10 bytes in the command message)
828:D          packed record
829:D              setunitvol: setunitvol_type;
830:D              nop: CMD_type;
831:D              setadd: CMD_type;
832:D              tva: tva_type;
833:D          end;
834:D      -10 begin (set_3V_address)
835:C          s3va.setunitvol := suv_CMD_pair(uep);
836:C          s3va.nop := CMDNo_op;
837:C          s3va.setadd := CMDset_address_3V;
838:C          s3va.tva.cylin := cylin;
839:C          s3va.tva.head := head;
840:C          s3va.tva.sect := sect;
841:C          HP18short_msge_out(uep, command_sec, addr(s3va), sizeof(s3va));
842:C          HP18wait_for_ppol(uep);
843:C          set_3V_address := qstat(uep);
844:C      2 end; (set_3V_address)
845:S
846:D
847:D      1 function spare_block(uep: uep_type; sparingmode: unsgn8): unsgn8;
848:S      (
849:S          issue the following command sequence:
850:S              SET_UNIT
851:S              SET_VOLUME
852:S              SPARE_BLOCK
853:S          return the QSTAT byte
854:S      )
855:D      var
856:D          sb: (the 4 bytes in the command message)
857:D          packed record
858:D              setunitvol: setunitvol_type;
859:D              spareblock: CMD_type;
860:D              sparingmode: unsgn8;
861:D          end;
862:D      -4 begin (spare_block)
863:C          sb.setunitvol := suv_CMD_pair(uep);
864:C          sb.spareblock := CMDspare_block;
865:C          sb.sparingmode := sparingmode;
866:C          HP18short_msge_out(uep, command_sec, addr(sb), sizeof(sb));
867:C          HP18wait_for_ppol(uep);
868:C          spare_block := qstat(uep);
869:C      2 end; (spare_block)
870:S
871:S
872:C      1 end; (CS801r)

```

```

873:D      1 $page$
874:S
875:D      1 module Qminit;
876:S
877:D      1 import
878:D      1 sysglobals, bkgnd, discHP18, CS80, CS80dsr, CS80ir, midecs;
879:S
880:D      1 export
881:D      1 function QdeviceName(uep: uep_type; var saved_ioresult: integer): string255;
882:D      1 function Qintdata(uep: uep_type): interleave_data;
883:D      1 function Qphydata(uep: uep_type): physical_data;
884:D      1 procedure Qgetinitparms;
885:D      1 procedure Qminitalize(uep: uep_type; interleave_factor: shortint);
886:S
887:D      1 implement {Qminit}
888:S
889:S
890:D      1 var
891:D      -38 1 describe_bytes: describe_type;
892:D      -40 1 formatting_option_byte: unsgn8;
893:D      -41 1 formatting_option_supported: boolean;
894:D      -42 1 diagnostic_to_be_rmt: boolean;
895:D      -44 1 initialize_options_byte: unsgn8;
896:D      -46 1 volume_ert_passes: unsgn8;
897:D      -48 1 clear_logs_option_byte: unsgn8;
898:S
899:S
900:D      -48 1 const
901:D      -48 1 default_initialize_options_byte = 0;
902:D      -48 1 default_volume_ert_passes = 2;
903:D      -48 1 default_clear_logs_option_byte = 1; {clear ert logs only}
904:S
905:S
906:D      1 function Qintdata(uep: uep_type): interleave_data;
907:C      begin (Qintdata)
908:C          with describe_bytes do
909:C              begin
910:C                  Qintdata.min := 1; {minimum}
911:C                  if formatting_option_supported
912:C                      then Qintdata.max := 255 {maximum subject to format specified}
913:C                      else Qintdata.max := mif; {true maximum}
914:C                  Qintdata.def := currentif; {default}
915:C              end; {with}
916:C      2 end; (Qintdata)
917:S
918:S
919:D      1 function Qphydata(uep: uep_type): physical_data;
920:C      begin (Qphydata)
921:C          with describe_bytes do
922:C              begin
923:C                  Qphydata.nspm := maxhadd+1; {number of surfaces per medium}
924:C                  Qphydata.ntps := maxcadd+1; {number of tracks/surface}
925:C                  if maxsadd>0 {number of sectors per track}
926:C                      then Qphydata.nspt := maxsadd+1 {actual}
927:C                      else Qphydata.nspt := (maxsvadd.1fb+1)*nbpb div 256; {pseudo}
928:C              end; {with}
929:C      2 end; (Qphydata)

```



```

930:D -48 1 $page$
931:S
932:D 1 function Qdevicename(uep: uep_type; var saved_ioresult: integer): string255;
933:S
934:D 2 var
935:D 2     retry_required: boolean;
936:D 2     devicename: string255;
937:D 2     index: integer;
938:D 2     bcd_product_number: (within the describe bytes)
939:D 2     packed record case integer of
940:D 2         0: (dn: usgn24);
941:D 2         1: (bcd: packed array[1..6] of usgn4);
942:D 2     end;
943:D 2     product_number: integer;
944:D 2     fixed_vol_byte:
945:D 2     packed record case integer of
946:D 2         0: (byte: usgn8);
947:D 2         1: (bit: packed array[0..7] of boolean);
948:D 2     end;
949:S
950:C 2 begin (Qdevicename)
951:S
952:C 2     ioresult := ord(inoerror);
953:S
954:C 2     try
955:C 3         allocate_bkgnd_info(uep);
956:S
957:C 3         if HPI8amigo_identify(uep) div 256 <> 2 then
958:C 4             ioresc_bkgnd(uep, znodvice);
959:S
960:C 3         repeat
961:C 4             retry_required := false;
962:C 4             if set_unitvol(uep) <> 0 then
963:C 5                 handle_bad_status(uep, true, retry_required);
964:C 4             until not retry_required;
965:S
966:C 3         repeat
967:C 4             retry_required := false;
968:C 4             if describe(uep, describe_bytes) <> 0 then
969:C 5                 handle_bad_status(uep, true, retry_required);
970:C 4             until not retry_required;
971:S
972:C 3         formatting_option_supported := false; (until proven otherwise)
973:C 3         if describe_bytes.ct.subset80 then (see if multiple formats supported)
974:C 4             try
975:C 5                 formatting_option_byte := 255; (test value)
976:C 5                 repeat
977:C 6                     retry_required := false;
978:C 6                     if formatting_option(uep, formatting_option_byte) <> 0 then
979:C 7                         handle_bad_status(uep, true, retry_required);
980:C 6                     until not retry_required;
981:C 6                     formatting_option_supported := true;
982:C 6                     formatting_option_byte := 0; (device's primary format)
983:C 5                 recover
984:C 5                 with bip_type(uep^.dvrtmp) do (confirm the command rejection)
985:C 6                     begin
986:C 6                         if (escapecode <> -10) or (ioresc <> zbadmode) then
987:C 7                             escape(escapecode);
988:C 6                         iores := inoerror;
989:C 6                     end; (recover)

```

```

990:S
991:C 3     deallocate_bkgnd_info(uep);
992:C 3     recover
993:C 3     begin
994:C 3         deallocate_bkgnd_info(uep);
995:C 3         if escapecode <> -10 then escape(escapecode);
996:C 3         ioresult := uep^.dvrtmp; (flag problem with describe)
997:C 3         if saved_ioresult = ord(inoerror) then (report to miui)
998:C 4             saved_ioresult := ioresult;
999:C 3         end; (recover)
1000:S
1001:C 2     if ioresult = ord(inoerror) then
1002:C 3         with describe_bytes do
1003:C 4             begin
1004:S
1005:C 4                 bcd_product_number.dn := dn;
1006:C 4                 product_number := 0;
1007:C 4                 for index := 1 to 5 do
1008:C 5                     product_number := product_number*10 + bcd_product_number.bcd[index];
1009:S
1010:C 4                 devicename := 'HP';
1011:C 4                 strwrite(devicename, 3, index, product_number:1);
1012:S
1013:C 4                 case dt of
1014:C 5                     0:
1015:C 5                         begin
1016:C 5                             fixed_vol_byte.byte := describe_bytes.fvb;
1017:C 5                             if fixed_vol_byte.bit[7-uep^.dv]
1018:C 6                                 then devicename := devicename + ' fixed disc'
1019:C 6                                 else devicename := devicename + ' removeable disc';
1020:C 5                             end;
1021:C 5                             devicename := devicename + ' tape';
1022:C 5                         otherwise (can't further describe the generic device type);
1023:S
1024:C 5                     end; (case)
1025:C 4                 end (with)
1026:C 4             else
1027:C 3                 devicename := '<inaccessible C580 device>';
1028:S
1029:C 2         Qdevicename := devicename;
1030:S
1031:C 2     end; (Qdevicename)

```



```

1032:D -48 1 $pages$
1033:S
1034:D -256 1 procedure allow_modification(prompt: string255; var parm: usgn8; maxparm: usgn8);
1035:D -256 2 var
1036:D -512 2 response: string255;
1037:D -520 2 newparm, i: integer;
1038:D -521 2 modification_ok: boolean;
1039:C 2 begin (allow_modification)
1040:C 2 repeat
1041:C 3 write(prompt, ' (defaults to ', parm:1, ') ');
1042:C 3 readln(response);
1043:C 3 modification_ok := strlen(response)=0;
1044:C 3 if not modification_ok then
1045:C 4 try
1046:C 5 strread(response, 1, 1, newparm);
1047:C 5 modification_ok := (i= strlen(response)+1) and (newparm>=0) and (newparm<=maxparm);
1048:C 5 if modification_ok then
1049:C 6 parm := newparm;
1050:C 5 recover
1051:C 5 if not {-escapecode in [8, 10]} then escape(escapecode);
1052:C 3 until modification_ok;
1053:C 2 end; (allow_modification)
1054:S
1055:S
1056:D 1 procedure Qgetinitparms;
1057:D 2 var
1058:D -1 ch: char;
1059:C 2 begin (Qgetinitparms)
1060:C 2 with describe_bytes do
1061:C 3 begin
1062:S 3 diagnostic_to_be_run := not ct.subset80 and (dt<=1); {C580 discs only}
1063:C 3 initialize_options_byte := default_initialize_options_byte;
1064:C 3 volume_ert_passes := default_volume_ert_passes;
1065:C 3 clear_logs_option_byte := default_clear_logs_option_byte;
1066:C 3
1067:S 3
1068:C 3 if extended_features_mode then
1069:C 4 begin (extended_features_mode stuff)
1070:C 4 writeln;
1071:C 4 if diagnostic_to_be_run then
1072:C 5 diagnostic_to_be_run := not yes('Suppress running the initial diagnostic?');
1073:C 4 allow_modification('Initialize options byte?', initialize_options_byte, 255);
1074:C 4 if not ct.subset80 and (dt<=1) then
1075:C 5 begin
1076:C 5 allow_modification('Volume ert passes?', volume_ert_passes, 255);
1077:C 5 allow_modification('Clear logs option byte?', clear_logs_option_byte, 255);
1078:C 5 end; (if)
1079:C 4 end (extended_features_mode stuff)
1080:C 4 else if dt=2 then {it's a tape; user may want to force re-certification}
1081:C 5 begin
1082:C 5 writeln;
1083:C 5 writeln('Medium is a tape: do you wish to force?');
1084:C 5 if yes(' re-certification if already certified?') then
1085:C 6 initialize_options_byte := 1; {force re-certification}
1086:C 5 end; (if)
1087:S 3
1088:C 3 if formatting_option_supported then
1089:C 4 begin
1090:C 4 writeln;
1091:C 4 allow_modification('Formatting option?', formatting_option_byte, 239);

```

```

1092:C 4 end; (if)
1093:S
1094:C 3 end; (with)
1095:C 2 end; (Qgetinitparms)

```



```

1096:D -48 1 $page$
1097:S
1098:D 1 procedure Qminimize(uep: uep_type; interleave_factor: shortint);
1099:S
1100:D 2 var
1101:D -1 2 retry_required: boolean;
1102:D -4 2 diag_unit: usgn4;
1103:D -5 2 ert_msge: boolean;
1104:D -8 2 de_log: de_log_type;
1105:D -10 2 head: usgn8;
1106:D -1080 2 log_bytes: log_info_type;
1107:D -1082 2 log_index: usgn8;
1108:D -1086 2 this_entry_ptr: ^log_entry_type;
1109:D -1087 2 sparing_required: boolean;
1110:D -1080 2 sparing_tries: signed16;
1111:S
1112:D -1090 2 const
1113:D -1090 2 sector_ert_passes = 20; {suspect sector scans}
1114:D -1090 2 track_ert_passes = 20; {spared track scans}
1115:D -1090 2 max_sparing_tries = 5; {maximum attempts to spare a sector/track}
1116:S
1117:C 2 begin {Qminimize}
1118:S
1119:C 2 try
1120:C 3 allocate_bkgnd_info(uep);
1121:C 3
1122:C 3 if diagnostic_to_be_run then
1123:C 4 begin
1124:C 4 diag_unit := uep^.du; {try the explicit unit; may get illegal opcode}
1125:C 4 repeat
1126:C 5 try
1127:C 6 repeat {make one pass of the complete diagnostic}
1128:C 7 retry_required := false;
1129:C 7 if initiate_diagnostic(uep, diag_unit, (loops) 1, (section) 0) <> 0 then
1130:C 8 handle_bad_status(uep, true, retry_required);
1131:C 7 until not retry_required;
1132:C 6 recover
1133:C 6 with bip_type(uep^.dvtemp) do {confirm the illegal opcode}
1134:C 7 if (escapecode <> 10) or (ioresc <> zbadmode) then
1135:C 8 escape(escapecode) {probably diagnostic result}
1136:C 8 else if diag_unit = 15 then {unit 15 was rejected!!!}
1137:C 9 ioresc_bkgnd(uep, zcatchall)
1138:C 9 else {explicit unit diagnostic not supported; use unit 15}
1139:C 9 begin
1140:C 9 iores := inoerror;
1141:C 9 diag_unit := 15; {original CS/80 allows unit 15 only}
1142:C 9 retry_required := true;
1143:C 9 end; {if}
1144:C 5 until not retry_required;
1145:C 4 end; {if}
1146:S
1147:C 3 if formatting_option_supported then
1148:C 4 repeat {set the user-specified formatting option}
1149:C 5 retry_required := false;
1150:C 5 if formatting_option(uep, formatting_option_byte) <> 0 then
1151:C 6 handle_bad_status(uep, true, retry_required);
1152:C 5 until not retry_required;
1153:S
1154:C 3 invalidate_stateinfo(uep); {formatting may change the blocksize!}
1155:S

```

```

1156:C 3 repeat {initialize the medium}
1157:C 4 retry_required := false;
1158:C 4 if initialize_media(uep, initialize_options_byte, interleave_factor) <> 0 then
1159:C 5 handle_bad_status(uep, true, retry_required);
1160:C 4 until not retry_required;
1161:S
1162:C 3 if not describe_bytes.ct.subset80 and (describe_bytes.dt <= 1) then
1163:C 4 begin {CS80 disc; perform testing and sparing}
1164:S
1165:C 4 repeat {preset the drive to force logging of runtime data errors}
1166:C 5 retry_required := false;
1167:C 5 if preset_drive(uep) <> 0 then
1168:C 6 handle_bad_status(uep, true, retry_required);
1169:C 5 until not retry_required;
1170:S
1171:C 4 repeat {clear the ert logs}
1172:C 5 retry_required := false;
1173:C 5 if clear_logs(uep, clear_logs_option_byte) <> 0 then
1174:C 6 handle_bad_status(uep, true, retry_required);
1175:C 5 until not retry_required;
1176:S
1177:C 4 repeat {set address 0}
1178:C 5 retry_required := false;
1179:C 5 if set_3v_address(uep, (cyl) 0, (head) 0, (sect) 0) <> 0 then
1180:C 6 handle_bad_status(uep, true, retry_required);
1181:C 5 until not retry_required;
1182:S
1183:C 4 if volume_ert_passes > 0 then
1184:C 5 repeat {run a complete volume ert the specified number of passes}
1185:C 6 retry_required := false;
1186:C 6 ert_msge := false; {do not request an execution message}
1187:C 6 if pattern_ert(uep, volume_ert, volume_ert_passes, ert_msge, log_bytes.entry[1]) <> 0 then
1188:C 7 handle_bad_status(uep, true, retry_required);
1189:C 6 until not retry_required;
1190:S
1191:C 4 for de_log := ert to runtime do
1192:C 5 for head := 0 to describe_bytes.maxhead do
1193:C 6 begin
1194:C 6 repeat {read the appropriate data error log}
1195:C 7 retry_required := false;
1196:C 7 if data_error_log(uep, de_log, head, log_bytes) <> 0 then
1197:C 8 handle_bad_status(uep, true, retry_required);
1198:C 7 until not retry_required;
1199:C 6
1200:C 6 for log_index := 1 to log_bytes.log_entries do
1201:C 7 begin
1202:C 7 this_entry_ptr := addr(log_bytes.entry[log_index]);
1203:C 7 with this_entry_ptr do
1204:C 8 begin
1205:C 8 repeat {set address to the suspect sector}
1206:C 9 retry_required := false;
1207:C 9 if set_3v_address(uep, logical_cyl, logical_head, logical_sect) <> 0 then
1208:C 10 handle_bad_status(uep, true, retry_required);
1209:C 9 until not retry_required;
1210:C 8
1211:C 8 repeat {run a sector ert; a message back implies sparing required}
1212:C 9 retry_required := false;
1213:C 9 sparing_required := true; {request an execution message}
1214:C 8
1215:C 8

```



```

1216:C      9      if pattern_err(uep, sector_err, sector_err_passes,
1217:C      10      sparing_required, this_entry_ptr^)<>0 then
1218:C      10      handle_bad_status(uep, true, retry_required);
1219:C      9      until not retry_required;
1220:S
1221:C      8      sparing_tries := 0;
1222:C      8      while sparing_required and (sparing_tries<max_sparing_tries) do
1223:C      9      begin
1224:C      9      sparing_tries := sparing_tries+1;
1225:S
1226:C      9      repeat {address the bad sector; may be different after the track err!}
1227:C      10      retry_required := false;
1228:C      10      if set_3V_address(uep, logical_cyl, logical_head, logical_sect)<>0 then
1229:C      11      handle_bad_status(uep, true, retry_required);
1230:C      10      until not retry_required;
1231:S
1232:C      9      repeat {spare without retaining data}
1233:C      10      retry_required := false;
1234:C      10      if spare_block(uep, (sparingmode) 1)<>0 then
1235:C      11      handle_bad_status(uep, true, retry_required);
1236:C      10      until not retry_required;
1237:S
1238:C      9      repeat {set address to the beginning of the affected track}
1239:C      10      retry_required := false;
1240:C      10      if set_3V_address(uep, logical_cyl, logical_head, 0)<>0 then
1241:C      11      handle_bad_status(uep, true, retry_required);
1242:C      10      until not retry_required;
1243:S
1244:C      9      repeat {run a track err; a message back implies sparing required again!!!}
1245:C      10      retry_required := false;
1246:C      10      sparing_required := true; {request an execution message}
1247:C      10      if pattern_err(uep, track_err, track_err_passes,
1248:C      11      this_entry_ptr^)<>0 then
1249:C      11      handle_bad_status(uep, true, retry_required);
1250:C      10      until not retry_required;
1251:S
1252:C      9      end; {while}
1253:S
1254:C      8      if sparing_required then ioresc_bkgnd(uep, zinitfail);
1255:S
1256:C      8      end; {with this_entry_ptr^}
1257:S
1258:C      7      end; {for log_index}
1259:S
1260:C      6      end; {for head}
1261:S
1262:C      4      end; {if device is a CS/80 disc}

```

```

1263:C      3 $page$
1264:S
1265:C      3      repeat {describe volume again to update the media parameters}
1266:C      4      retry_required := false;
1267:C      4      if describe(uep, describe_bytes)<>0 then
1268:C      5      handle_bad_status(uep, true, retry_required);
1269:C      4      until not retry_required;
1270:S
1271:C      3      deallocate_bkgnd_info(uep),
1272:S
1273:C      3      recover
1274:C      3      begin
1275:C      3      abort_bkgnd_process(uep);
1276:C      3      ioresult := uep^.dvrtemp;
1277:C      3      uep^.dvrtemp := ord(ioreerror); {report the error only once}
1278:C      3      escape(-10);
1279:C      3      end; {recover}
1280:S
1281:C      2 end; {Qmitalize}
1282:S
1283:C      1 end {Qminit}
1284:S
1285:C      2 $include 'QMINIT'$;

```

No errors. No warnings.

***** Nonstandard language features enabled *****

Description

MIUI is the “mediainit user interface” and contains the device-independent portion of mediainit.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:D 0 $modcal, stackcheck off, ioccheck off, debug off, range off, ovflcheck off$
2:D 0
3:D 0 $search 'IOLIB:KERNEL', 'INIT:DRVASH', 'INIT:D1SCHPI8',
4:D 0 'INIT:AMIGO', 'INIT:CS80', 'FMINIT', 'MINIT'$
5:S
6:S
7:S
8:D 0 program miui(input,output,listing); (mediainit user interface program)
9:S
10:D 1 import
11:D 1 sysglobals, misc, sysdevs, fs, iodeclarations, midecs, bkgnd,
12:D 1 Mminit, Hminit, Fminit, Xminit, Qminit, Bminit;
13:S
14:D 1 type
15:D 1 sup_dev_type = (M,N,U,V,W,H,F,C,D,P,X,8,Q); (supported devices)
16:S
17:D 1 var
18:D -4 1 lun: integer;
19:D -8 1 uep: uep_type;
20:D -10 1 device: sup_dev_type;
21:S
22:S
23:D -256 1 procedure fatal_ioreult(message: string255; iores: integer);
24:C 2 begin {fatal_ioreult}
25:C 2 writein(message);
26:C 2 getioerrmsg(message, iores);
27:C 2 writein(' ');
28:C 2 escape(-1);
29:C 2 end; {fatal_ioreult}
30:S
31:S
32:D 1 function on_same_medium(flun: unitnum): boolean;
33:C 2 begin {on_same_medium}
34:C 2 with unitable^[flun] do
35:C 3 on_same_medium := (sc=uep^.sc) and (ba=uep^.ba) and
36:C 3 (du=uep^.du) and (dv=uep^.dv) and
37:C 3 (letter=uep^.letter);
38:C 2 end; {on_same_medium}

```

```

39:D -10 1 $page$
40:S
41:D 1 procedure write_program_header;
42:S
43:D 2 const
44:D 2 previd = daterec[year:84, day:4, month:6];
45:D 2 prevno = '3.0';
46:D 2 programname = 'Mediainit'; (title)
47:D 2 copyright1 = ' Copyright 1984 Hewlett-Packard Company.';
48:D 2 copyright2 = ' All rights reserved.';
49:D 2 monthstr = 'JanFebMarAprMayJunJulAugSepOctNovDec';
50:S
51:D 2 var
52:D -2 2 date: daterec;
53:D -6 2 time: timerec;
54:D -10 2 i: integer;
55:D -14 2 seconds: integer;
56:D -24 2 todaysdate: string[9];
57:D -34 2 timestring: string[8];
58:S
59:C 2 begin {write_program_header}
60:C 2
61:C 2 sysdate(date);
62:C 2 systime(time);
63:C 2
64:C 2 with date do
65:C 3 if (month in [1..12]) and (day>0) and (year<100) then
66:C 4 begin
67:C 4 todaysdate := ' '+str(monthstr, month*3-2, 3)+'- ';
68:C 4 if day>9 then todaysdate[1] := chr(day div 10 + ord('0'));
69:C 4 todaysdate[2] := chr(day mod 10 + ord('0'));
70:C 4 todaysdate[8] := chr(year div 10 + ord('0'));
71:C 4 todaysdate[9] := chr(year mod 10 + ord('0'));
72:C 4 end {then}
73:C 4 else
74:C 4 todaysdate := ' ';
75:C 2
76:C 2 setstrlen(timestring,0);
77:C 2 i := 1;
78:C 2 if time.hour<10 then
79:C 3 strwrite(timestring,i,1,'0');
80:C 3 strwrite(timestring,i,1,time.hour:1,'');
81:C 2 if time.minute<10 then
82:C 3 strwrite(timestring,i,1,'0');
83:C 3 strwrite(timestring,i,1,time.minute:1,'');
84:C 2 seconds := time.centisecond div 100;
85:C 2 if seconds<10 then
86:C 3 strwrite(timestring,i,1,'0');
87:C 3 strwrite(timestring,i,1,seconds:1);
88:S
89:C 2 writein(programname, ' [Rev ',prevno,' ',previd,month:2,'/',previd.day:2,
90:C 2 '/',previd.year:2,']',todaysdate:10,' ',timestring);
91:C 2 writein;
92:C 2 writein(copyright1);
93:C 2 writein(copyright2);
94:C 2 writein;
95:S
96:C 2 end; {write_program_header}

```



```

97:D -10 1 $page$
98:D 1 procedure request_vol;
99:D 2
100:D 2 type
101:D 2 string27 = string[27];
102:D 2 dev_name_type = array [M..B] of string27; {rdq}
103:D 2
104:S 2
105:D 2 const
106:D 2 dev_name = (device names)
107:D 2 dev_name_type
108:D 2 [
109:D 2 (M) 'Internal minifloppy',
110:D 2 (N) 'HP8290X series minifloppy',
111:D 2 (U) 'HP913XA series hard disc',
112:D 2 (V) 'HP913XB series hard disc',
113:D 2 (W) 'HP913XC series hard disc',
114:D 2 (H) 'HP988S flexible disc',
115:D 2 (F) 'HP988S flexible disc',
116:D 2 (C) 'HP7906 removable hard disc',
117:D 2 (D) 'HP7906 fixed hard disc',
118:D 2 (P) 'HP7920 hard disc',
119:D 2 (X) 'HP7925 hard disc',
120:D 2 (B) 'HP98259 bubble memory'; {rdq}
121:D 2 nonabortive_ioresult_set = [ord(inoerror), ord(zbadblock), ord(znoblock),
122:D 2 ord(zuninitialized), ord(inodirectory)];
123:D 2
124:D 2 var
125:D 2 response: string255;
126:D 2 fvid: vid;
127:D 2 ftitle: fid;
128:D 2 fsegs: integer;
129:D 2 fkind: filekind;
130:D 2 scantitle_ok: boolean;
131:D 2 saved_ioresult: integer;
132:D 2 devicename: string255;
133:C 2
134:S 2 begin {request_vol}
135:C 2
136:C 2 repeat
137:C 2 write('volume ID? ');
138:C 2 readln(response);
139:C 2 if (strlen(response)=0) or (strpos(#27,response)<>0) then escape(-1);
140:C 2 scantitle_ok := scantitle(response, fvid, ftitle, fsegs, fkind);
141:C 2 if scantitle_ok and (ftitle='') then
142:C 2 begin
143:C 2 lun := findvolume(fvid, true);
144:C 2 saved_ioresult := ioresult; {for decoding below}
145:C 2 getioerrmsg(response, ord(inounit)); {for the case lun=0}
146:C 2 end {then}
147:C 2 else
148:C 2 begin
149:C 2 lun := 0;
150:C 2 response := 'illegal syntax';
151:C 2 end; {else}
152:C 2 if lun=0 then
153:C 2 writeln(' ', response);
154:C 2 until lun<>0;
155:C 2 extended_features_mode := fsegs<>0;

```

```

156:C 2 if extended_features_mode then
157:C 2 begin
158:C 2 writeln;
159:C 2 writeln('*** Warning ***');
160:C 2 writeln(' You have invoked an "extended features mode,"');
161:C 2 writeln(' which for certain devices requires you to have');
162:C 2 writeln(' extensive knowledge about the device's command');
163:C 2 if not yes(' set. Are you SURE you want to proceed?') then
164:C 2 escape(-1);
165:C 2 end; {if}
166:S 2
167:C 2 uep := addr(untable^[lun]);
168:C 2 with uep do
169:C 2 begin
170:C 2 case letter of
171:C 2 'M': device := M;
172:C 2 'N': device := N;
173:C 2 'U': device := U;
174:C 2 'V': device := V;
175:C 2 'W': device := W;
176:C 2 'H': device := H;
177:C 2 'F': device := F;
178:C 2 'C': if dv=0 then device := C;
179:C 2 else device := D;
180:C 2 'P': device := P;
181:C 2 'X': device := X;
182:C 2 'B': device := B; {rdq}
183:C 2 'Q': device := Q;
184:C 2 otherwise
185:C 2 begin
186:C 2 writeln('Mediainit does not support this device');
187:C 2 escape(-1);
188:C 2 end;
189:C 2 end; {case}
190:S 2
191:C 2 if device=Q
192:C 2 then devicename := Qdevicename(uep, saved_ioresult) {may do I/O, alter saved_ioresult}
193:C 2 else devicename := dev_name[device];
194:C 2 writeln;
195:C 2 write('Device: ', devicename);
196:S 2
197:C 2 if device<>M then {it has a select code & possibly a bus address}
198:C 2 if (device=F) or (device=B) {it has a select code only} {rdq}
199:C 2 then write(' ', sc:1);
200:C 2 else write(' ', 100*sc+ba:1);
201:C 2 if device<>B then write(' ', dv:1); {rdq}
202:C 2 if device in [C,D,Q] then write(' ', dv:1);
203:C 2 writeln;
204:C 2 end; {with uep}
205:S 2
206:C 2 write('logical unit #', lun:1, ' - ');
207:C 2 if fvid[1]<>'#' then writeln(' ', fvid:1); {rdq}
208:C 2 else {rdq}
209:C 2 if (saved_ioresult in nonabortive_ioresult_set) or {rdq}
210:C 2 ((device=B) and (saved_ioresult<>ord(znodevice))) and {rdq}
211:C 2 (saved_ioresult<>ord(zbadhardware))) {rdq}
212:C 2 then writeln('no directory') {rdq}
213:C 2 else
214:C 2 fatal_ioresult('Mediainit aborted', saved_ioresult);
215:S 2

```



```
216:  2  end; {request_vol}
```

```
217:  -10 1 $page$
218:  1 procedure other_volume_check;
219:  2 var
220:  2   warning_issued: boolean;
221:  -1 2   flun: unitnum;
222:  -4 2   line: shortint;
223:  -6 2   CRline, CRTcolumn, stopline: integer;
224:  -18 2   fvid: vid;
225:  -36 2   const
226:  2     maxlines = 10;
227:  -36 2     fieldwidth = 16;
228:  -36 2   begin {other_volume_check}
229:  2     warning_issued := false;
230:  2     for flun := 1 to maxunit do
231:  3       if on_same_medium(flun) then
232:  4         with unitable^[flun] do
233:  5           begin
234:  6             if flun <> lun then
235:  7               begin
236:  8                 if not warning_issued then
237:  9                   writeln('WARNING: the initialization will also destroy:');
238:  7                   warning_issued := true;
239:  7                   line := 1;
240:  7                   end; {if}
241:  6                   fgotoxy(output, CRTcolumn, CRTline);
242:  6                   write('#', flun:1);
243:  6                   fgotoxy(output, CRTcolumn+5, CRTline);
244:  6                   call(dam, fvid, flun, getvolumename);
245:  6                   if strlen(fvid) > 0
246:  7                     then write(fvid, ':')
247:  7                     else write('<no dir>');
248:  6                   if CRTcolumn = 0
249:  7                     then writeln
250:  7                       (scrolls the screen if necessary)
251:  7                     else fgotoxy(output, CRTcolumn, CRTline+1);
252:  6                   line := line+1;
253:  6                   if line > maxlines then
254:  7                     begin
255:  8                       fgotoxy(output, CRTcolumn, stopline);
256:  8                       fgotoxy(output, CRTcolumn+fieldwidth, stopline-maxlines);
257:  7                       line := 1;
258:  7                       end; {if}
259:  6                   end; {if flun <> lun}
260:  5                   umevalid := false;
261:  5                   end; {with}
262:  2   if warning_issued and (CRTcolumn <> 0) then
263:  3     fgotoxy(output, 0, stopline);
264:  2   end; {other_volume_check}
```



```

273:D -10 1 $page$
274:S
275:D 1 procedure user_confirmation;
276:C 2 begin {user_confirmation}
277:C 2 writeln;
278:C 2 if not yes('Are you SURE you want to proceed?') then
279:C 2 escape(-1);
280:C 2 end; {user_confirmation}
281:S
282:S
283:D 1 procedure initialize_medium;
284:S
285:D 2 var
286:D 2 interleave: shortint;
287:S
288:D -6 2 procedure usi(intdata: interleave_data);
289:D -6 3 var
290:D -262 3 response: string255;
291:D -266 3 i: integer;
292:D -267 3 number_conversion_ok: boolean;
293:C 3 begin {user-specified interleave}
294:C 3 with intdata do
295:C 4 if (def>=min) and (def<=max) and (min<max) then
296:C 5 begin
297:C 5 writeln;
298:C 5 repeat
299:C 6 write('Interleave factor? ');
300:C 6 if max<>255 then
301:C 6 write([' ', min:1, ' ', max:1, ' ']);
302:C 6 write(['(defaults to ', def:1, ' ')']);
303:C 6 readln(response);
304:C 6 number_conversion_ok := strlen(response)=0;
305:C 6 if number_conversion_ok then
306:C 7 interleave := def;
307:C 7 else
308:C 7 try
309:C 8 strread(response, 1, i, interleave);
310:C 8 number_conversion_ok := i= strlen(response)+1;
311:C 8 recover
312:C 8 if not (-escapecode in [8, 10]) then escape(escapecode);
313:C 8 until number_conversion_ok and (interleave>=min) and (interleave<=max);
314:C 8 end {then}
315:C 8 else
316:C 8 interleave := def;
317:C 3 end; {user-specified interleave}
318:S
319:D 2 function fudged_d(dev: sup_dev_type; letter: char): char;
320:C 3 begin {fudged_d}
321:C 3 if dev=0
322:C 4 then fudged_d := 'D'
323:C 4 else fudged_d := letter;
324:C 3 end; {fudged_d}

```

```

325:D -2 2 $page$
326:S
327:C 2 begin {initialize_medium}
328:C 2 { determine interleave factor }
329:C 2 case device of
330:C 3 M: usi(Mintdata);
331:C 3 N: usi(Nintdata(uep));
332:C 3 U,V,W: interleave := 9; {N/A but passed as a parameter}
333:C 3 H: usi(Hintdata);
334:C 3 F: usi(Fintdata);
335:C 3 Q: begin
336:C 3 Qgetinitparms;
337:C 3 usi(Qintdata(uep));
338:C 3 end;
339:C 3 otherwise {not applicable};
340:C 3 end; {case}
341:S
342:C 2 lockup;
343:C 2 try
344:C 3 writeln;
345:C 3 writeln('Medium initialization in progress');
346:C 3 with uep do
347:C 4 case device of
348:C 5 M: Hinitialize(interleave);
349:C 5 H,N,U,V,W: Hinitialize(uep, interleave);
350:C 5 F: Finitialize(isc_table[sc].card_ptr, du, interleave);
351:C 5 C,D,P,X: Xinitialize(fudged_d(device, letter), sc, ba, du);
352:C 5 Q: Qinitialize(uep, interleave);
353:C 5 B: Binitialize(lun); {rdq}
354:C 3 end; {case}
355:C 3 writeln('Medium initialization completed');
356:C 3 escape(0); {to do the lockdown}
357:C 3 recover
358:C 3 begin
359:C 3 lockdown;
360:C 3 if escapecode=0 then
361:C 4 {do nothing}
362:C 4 else if escapecode=-10 then
363:C 4 fatal_ioresult('#7'Medium initialization aborted:', ioresult)
364:C 4 else
365:C 5 escape(escapecode);
366:C 3 end; {recover}
367:S
368:C 2 end; {initialize_medium}
369:S
370:S
371:D 1 procedure make_initial_directories;
372:D 2
373:D 2 var
374:D -2 flun: unitnum;
375:D -82 dircatentry: catentry;
376:D -744 dam_fib: fib;
377:D -748 j: integer;
378:S
379:D 2 procedure check_ioresult;
380:C 3 begin
381:C 3 if ioresult<>ord(inoerror) then
382:C 4 fatal_ioresult('#7'Volume zeroing aborted:', ioresult);
383:C 3 end;

```



```

384:D -748 2 $page$
385:S
386:D 2 procedure make_initial_LIF_directory;
387:S
388:D 3 type
389:D 3 bcd = 0..9;
390:D 3 bcd12 = packed array[1..12] of bcd;
391:D 3 b15 = 0..32767;
392:S
393:D 3 LIF_vol_type =
394:D 3 packed record
395:D 3 id: shortint;
396:D 3 vol_label: packed array[1..6] of char;
397:D 3 dir_sa: integer;
398:D 3 oct_10000: shortint;
399:D 3 dummy: shortint;
400:D 3 dir_len: integer;
401:D 3 version: shortint;
402:D 3 zero: shortint;
403:D 3 phydata: physical_data;
404:D 3 vol_stamp: bcd12;
405:D 3 end;
406:S
407:D 3 LIF_dir_entry =
408:D 3 packed record
409:D 3 file_name: packed array[1..10] of char;
410:D 3 file_type: shortint;
411:D 3 start_add: integer;
412:D 3 file_len: integer;
413:D 3 toc: bcd12;
414:D 3 l_flag: boolean;
415:D 3 vol_num: b15;
416:D 3 implem: integer;
417:D 3 end;
418:S
419:D 3 sector_type =
420:D 3 record case integer of
421:D 3 1: (LIF_vol_label: LIF_vol_type);
422:D 3 2: (LIF_directory: array[0..7] of LIF_dir_entry);
423:D 3 end;
424:S
425:D 3 const
426:D 3 null_bcd12 = bcd12[0,0,0,0,0,0,0,0,0,0,0,0];
427:S
428:D 3 P_LIF_v1 = (standard LIF volume label)
429:D 3 LIF_vol_type
430:D 3 [id: 32768,
431:D 3 vol_label: 'P9826 ',
432:D 3 dir_sa: 2,
433:D 3 oct_10000: 4096,
434:D 3 dummy: 0,
435:D 3 dir_len: 2,
436:D 3 version: 1,
437:D 3 zero: 0,
438:D 3 phydata: physical_data[ntps: 0, nsmp: 0, nspt: 0], (filled in later)
439:D 3 vol_stamp: null_bcd12 ];
440:S
441:D 3 P_LIF_lead = (logical end of directory mark)
442:D 3 LIF_dir_entry
443:D 3 [file_name: '

```

```

444:D 3 file_type: -1,
445:D 3 start_add: 0,
446:D 3 file_len: 0,
447:D 3 toc: null_bcd12,
448:D 3 l_flag: false,
449:D 3 vol_num: 0,
450:D 3 implem: 0 ];
451:S
452:D 3 var
453:D 3 buffer: (four sector buffer for LIF volume label and dummy directory)
454:D 3 record case integer of
455:D 3 0: (long: array[0..255] of integer);
456:D 3 1: (sector: array[0..3] of sector_type);
457:D 3 end;
458:D -1024 3 i: shortint;
459:D -1026 3 tm_fib: fib;
460:D -1688 3
461:C 3 begin (make_initial_LIF_directory)
462:S
463:C 3 with buffer do
464:C 4 begin
465:S
466:C 4 for i := 0 to 255 do (zero the entire buffer)
467:C 5 long[i] := 0;
468:S
469:C 4 with sector[0], LIF_vol_label do
470:C 5 begin
471:C 5 LIF_vol_label := P_LIF_v1;
472:C 5 case device of
473:C 6 M: phydata := Mphydata;
474:C 6 H,N,U,V,W,C,D,P,X: phydata := Hphydata(uep);
475:C 6 F: phydata := Fphydata;
476:C 6 Q: phydata := Qphydata(uep);
477:C 6 B: phydata := Bphydata(lun); (rdq)
478:C 6 end; (case)
479:C 5 end; (with)
480:S
481:C 4 sector[2].LIF_directory[0] := P_LIF_lead;
482:S
483:C 4 end; (with)
484:S
485:C 3 with tm_fib do
486:C 4 begin
487:C 4 funit := lun;
488:C 4 fileid := 0;
489:C 4 fpeof := sizeof(buffer);
490:C 4 end; (with)
491:S
492:C 3 uep^.ureportchange := false;
493:C 3 call(uep^.tm, addr(tm_fib), writebytes, buffer, sizeof(buffer), 0);
494:C 3 check_ioresult;
495:C 3 call(uep^.tm, addr(tm_fib), flush, buffer, 0, 0); (flush tapebuf)
496:C 3 check_ioresult;
497:C 3 call(uep^.tm, addr(tm_fib), clearunit, buffer, 0, 0); (update tape parms)
498:C 3 check_ioresult;
499:C 3 uep^.ureportchange := true;
500:S
501:C 3 end; (make_initial_LIF_directory)

```



```

502:D -748 2 $page$
503:S
504:C 2 begin {make_initial_directories}
505:S
506:C 2 writeln('Volume zeroing in progress');
507:C 2
508:S
509:C 2 if uep^.byteoffset=0 then
510:C 3 make_initial_LIF_directory;
511:S
512:C 2 for flun := maxunit downto 1 do
513:C 3 with unitable[flun] do
514:C 4 if (flun=lun) or (on_same_medium(flun) and (byteoffset<>uep^.byteoffset)) then
515:C 5 begin
516:C 6 with dircatentry do
517:C 6 begin
518:C 6 cname := 'V';
519:C 6 strwrite(cname, 2, j, flun:1);
520:C 6 ureportchange := false; {for ueovbytes' sake}
521:C 6 cpsize := ueovbytes(flun);
522:C 6 ureportchange := true;
523:C 6 cextral := 0; {let dam decide # of dir entries}
524:C 6 end; {with}
525:C 5 with dam_fib do
526:C 6 begin
527:C 6 funit := flun;
528:C 6 fwindow := addr(dircatentry);
529:C 6 call(dam, fvid, flun, getvolumename); {assign fvid}
530:C 6 pathid := 1;
531:C 6 ftitle := '';
532:C 6 end; {with}
533:C 5 call(dam, dam_fib, flun, makedirectory);
534:C 5 check_ioreult;
535:C 5 end; {then}
536:S
537:C 2 writeln('Volume zeroing completed');
538:S
539:C 2 end; {make_initial_directories}

```

```

540:D -10 1 $page$
541:S
542:C 1 begin {miui}
543:C 1 try
544:C 2 initialize_bkgnd;
545:C 2 write_program_header;
546:C 2 request_vol;
547:C 2 other_volume_check;
548:C 2 user_confirmation;
549:C 2 initialize_medium;
550:C 2 make_initial_directories;
551:C 2 recover
552:C 2 if escapecode=-20 then
553:C 3 begin
554:C 3 writeln;
555:C 3 writeln('Mediainit stopped by user');
556:C 3 end {then}
557:C 3 else
558:C 3 escape(escapecode);
559:C 1 end. {miui}
560:S
561:S

```

No errors. 1 warnings.

***** Nonstandard language features enabled *****

MOUSE

Description

MOUSE provides driver code for the mouse and other HPHIL relative locator input devices.

Requirements

SYSDEVS, and SYSGLOBALS.

Notes

HPHIL must be *executed* either before or after MOUSE executes for the mouse to be functional.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1984.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S     RESTRICTED RIGHTS LEGEND
11:S
12:S     Use, duplication, or disclosure by the Government
13:S     is subject to restrictions as set forth in
14:S     paragraph (b) (3) (8) of the Rights in Technical
15:S     Data and Computer Software clause in
16:S     DAR 7-104.9(a).
17:S
18:S     HEWLETT-PACKARD COMPANY
19:D     0 Fort Collins, Colorado      *)
20:S
21:D     0 $sysprogs
22:D     0 $stackcheck off$ $heap_dispose off$
23:D     0 $iocheck off$ $range off$ $ovflcheck off$
24:D     0 $debug off$
25:S
26:D     0 program mouseinit;
27:S
28:D     1 module mouse; { mouse module }
29:D     1 import sysdevs,sysglobals;
30:S
31:D     1 export
32:S
33:D     1 function initmouse:boolean;
34:S
35:D     1 implement
36:D     1 type
37:D     1   sint = -32768..32767;
38:S
39:D     1   pollblock = packed record
40:D     1     case integer of
41:D     1       0:(d: packed array[1..7] of byte);
42:D     1       1:(twosets : boolean;
43:D     1         kcodes   : 0..3;
44:D     1         filler   : 0..7;
45:D     1         numaxes  : 0..3;
46:D     1         xdata    : sint;
47:D     1         ydata    : sint;
48:D     1         zdata    : sint)
49:D     1   end;
50:D     1   lkeytype = packed record
51:D     1     case integer of
52:D     1       0:(b:byte);
53:D     1       1:(row:0..15;
54:D     1         col:0..7;
55:D     1         up : boolean)
56:D     1   end;
57:D     1   kmttype = array[0..7] of byte;
58:D     1 const
59:D     1   mkeymap = kmttype[3,13,8,0,0,0,0,0]; { ex,cr,bs,... }
60:D     1 var

```

```

61:D     -4 1   driver : loopdrvptr;
62:D     -12 1  keystate : array[0..7] of boolean;
63:D     -24 1  cumdata : array[1..3] of integer;
64:D     -32 1  gdata : pollblock;
65:D     -34 1  cindex : 0..3;
66:S
67:D     1 procedure doreset;
68:D     2 var i : sint;
69:D     2 begin
70:D     2   for i:=0 to 7 do keystate[i]:=true; { all keys up }
71:D     2   for i:=1 to 7 do gdata.d[i]:=0;
72:D     2   for i:=1 to 3 do cumdata[i]:=0;
73:D     2 end;
74:S
75:D     1 procedure mdataproc(var statbyte,databyte:byte; var done:boolean);
76:D     2 var
77:D     2   k : lkeytype;
78:D     2 begin {mdataproc: mouse data proc }
79:D     2   WITH LOOPCONTROL^ DO {4/9/84 SFB}
80:D     2   with loopdevices[loopdevice] do
81:D     2   case devstate of
82:D     2     1:begin
83:D     2       gdata.d[1]:=databyte; cindex:=0;
84:D     2       if gdata.numaxes=0 then devstate:=8 else devstate:=2;
85:D     2     end;
86:D     2     2,4,6:
87:D     2     begin { co-ord data } { low byte }
88:D     2       if not descrip.size16 then
89:D     2         if databyte>127 then gdata.d[devstate]:=255 { sign extend }
90:D     2         else gdata.d[devstate]:=0;
91:D     2       gdata.d[devstate+1]:=databyte;
92:D     2       devstate:=devstate+1+ord(not descrip.size16);
93:D     2       if not descrip.size16 then
94:D     2       begin { 8 bit mode }
95:D     2         cindex:=cindex+1;
96:D     2         if cindex>=gdata.numaxes then devstate:=8;
97:D     2       end;
98:D     2     end;
99:D     2     3,5,7: { co-ord data } { high byte }
100:D     2     begin
101:D     2       gdata.d[devstate-1]:=databyte;
102:D     2       devstate:=devstate+1;
103:D     2       cindex:=cindex+1;
104:D     2       if cindex>=gdata.numaxes then devstate:=8;
105:D     2     end;
106:D     2     8:begin { keydata }
107:D     2       case gdata.kcodes of
108:D     2         0: { not supposed to be any data }
109:D     2         1:begin { is all ready ascii }
110:D     2           statbyte:=0;
111:D     2           call(kbdisrhook,statbyte,databyte,done);
112:D     2         end;
113:D     2       otherwise { key code data }
114:D     2       k.b:=databyte;
115:D     2       keystate[k.col]:=k.up; { record the state }
116:D     2       if not k.up then
117:D     2       begin
118:D     2         statbyte:=0; databyte:=mkeymap[k.col];
119:D     2         call(kbdisrhook,statbyte,databyte,done);
120:D     2       end;

```



```

121:C      5      end;
122:C      5      end;
123:C      5      end;
124:C      2      end; {mdataproc}
125:S
126:D      1      procedure mopsproc(op:loopdvrop);
127:D      2      var
128:D      -4 2      limit : integer;
129:D      -8 2      statbyte,databyte : byte;
130:D      -9 2      done : boolean;
131:S
132:D      2      procedure check(i:integer;d:sint);
133:C      3      begin
134:C      3      cumdata[i]:=cumdata[i]+d; done:=false;
135:C      3      if abs(cumdata[i])>limit then
136:C      4      begin
137:C      4      done:=true;
138:C      4      if (i=1) then
139:C      4      begin
140:C      5      statbyte:=hex('FF'); { normal } { x }
141:C      5      if d>0 then databyte:=255 else databyte:=1;
142:C      5      end
143:C      5      else
144:C      5      begin
145:C      5      statbyte:=hex('EF'); { shift } { y }
146:C      5      if d>0 then databyte:=1 else databyte:=255;
147:C      5      end;
148:C      4      cumdata[i]:=0;
149:C      4      end;
150:C      3      end; { check }
151:S
152:C      2      begin { mopsproc; mouse ops }
153:C      2      WITH LOOPCONTROL^ DO {4/9/84 SFB}
154:C      3      with loopdevices[loopdevice] do
155:C      4      case op of
156:C      5      datastarting: devstate:=1;
157:C      5      dataended:
158:C      5      begin
159:C      5      if descrip.size16 then limit:=descrip.counts div 8;
160:C      5      else limit:=descrip.counts div 800;
161:C      5      check(1,gdata.xdata);
162:C      5      if done then
163:C      5      begin call(rpgisrhook,statbyte,databyte,done);
164:C      5      gdata.ydata:=0;
165:C      5      end
166:C      5      else
167:C      5      begin
168:C      5      check(2,gdata.ydata);
169:C      5      if done then
170:C      5      begin call(rpgisrhook,statbyte,databyte,done);
171:C      5      gdata.xdata:=0;
172:C      5      end;
173:C      5      end;
174:C      5      end;
175:C      5      resetdevice: doreset;
176:C      5      end;
177:C      2      end; { mopsproc }
178:D      -34 1
179:D      1      function initmouse:boolean;
180:C      2      begin

```

```

181:C      2      if (kbdtype=itfkbd) and (driver=nil) then
182:C      3      begin
183:C      3      new(driver);
184:C      3      with driver do
185:C      4      begin { initialize driver log in record }
186:C      4      lowid:=96; { all relative pointers }
187:C      4      highid:=127;
188:C      4      daddr:=0; { any device address }
189:C      4      opsproc:=mopsproc; { set procedure vars }
190:C      4      dataproc:=mdataproc;
191:C      4      next:=loopdriverlist; { add to driver list }
192:C      4      loopdriverlist:=driver;
193:C      4      end;
194:C      3      if LOOPCONTROL^ loopisok {4/9/84 SFB}
195:C      3      then CALL(HPHILCPDRHOOK,CONFIGUREOP); {4/9/84 SFB}
196:C      3      initmouse:=true;
197:C      3      end
198:C      3      else initmouse:=false;
199:C      2      end; { initloop }
200:S
201:C      1      end; { mouse interface module }
202:S
203:D      1      import mouse, loader;
204:S
205:C      1      begin
206:C      1      if initmouse then markuser;
207:C      1      end.
208:S
209:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

NONUSKBD1

Description

NONUSKBD1 provides initialization for handling 98203 non-U.S.-language keyboards, and handles all Katakana keyboards.

Requirements

SYSDEVS, MISC, SYSGLOBALS, and ASM.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```
1: S (*
2: S
3: S
4: S (c) Copyright Hewlett-Packard Company, 1984.
5: S All rights are reserved. Copying or other
6: S reproduction of this program except for archival
7: S purposes is prohibited without the prior
8: S written consent of Hewlett-Packard Company.
9: S
10: S RESTRICTED RIGHTS LEGEND
11: S
12: S Use, duplication, or disclosure by the Government
13: S is subject to restrictions as set forth in
14: S paragraph (b) (3) (B) of the Rights in Technical
15: S Data and Computer Software clause in
16: S DAR 7-104.9(a).
17: S
18: S HEWLETT-PACKARD COMPANY
19: D 0 Fort Collins, Colorado *)
20: S
21: D 0 $SYSPROG$
22: D 0 $heap_dispose off$
23: D 0 $iocheck off$
24: D 0 $range off$ $ovflcheck off$
25: D 0 $debug off$
26: D 0 $stackcheck off$
27: D 0 program NONUS1INIT;
28: S
29: D 1 module NON_US_KBD1;
30: S
31: D 1 import sysdevs,misc;
32: D 1 export
33: D 1 function initlang:boolean;
34: S
35: D 1 implement
36: S
37: D 1 procedure katakanatrans;
38: D 2 label 1;
39: D 2 { katakana language keycode->character code translations }
40: S
41: C 2 begin {katakanatrans}
42: C 2 if langcom.status=0 then langcom.result:=alpha_key
43: C 3 else
44: C 3 if langcom.data>127 then langcom.result:=ignored_key
45: C 4 else
46: C 4 with langcom, langtable[0]^, keytable[data] do
47: C 5 begin
48: C 5 result := keyclass;
49: C 5 if (result=ignored_key) and not kbdaltlock then goto 1;
50: S
51: C 5 if (data=18) or (data=19) then { change to/from alternate }
52: C 6 begin result:=ignored_key; kbdaltlock:=(data=18); end
53: C 6 else
54: C 6 if control and (kbdtype<>ITFKBD) and
55: C 7 ((data=96) or (data=97)) then { change to/from alternate }
56: C 7 begin result:=ignored_key; kbdaltlock:=(data=96); end
57: C 7 else
58: C 7 if kbdaltlock then
59: C 8 begin { use alternate language }
60: C 8 langindex:=1-langindex; extension:=false;
```

```
61: C 8 call(langtable[langindex]^semantics);
62: C 8 langindex:=1-langindex;
63: C 8 end
64: C 8 else
65: C 8 if result<>ignored_key then
66: C 9 begin { normal processing }
67: C 9 extension := false;
68: C 9 shift:= shift and not no_shift;
69: C 9 key := keys[shift];
70: C 9 end;
71: C 9 { with langcom etc. }
72: C 1: end; { katakanatrans }
73: S
74: D 1 function initlang:boolean;
75: D 2 type
76: D 2 t1 = packed array[boolean, 60..98] of char;
77: D 2 t2 = packed array[FRENCH_KBD..KATAKANA_K80] of t1;
78: S
79: D 2 const
80: D 2 keylookup = t2[
81: D 2 { french }
82: D 2 t1[0..+123-456*789/E()^1234567890''#170#200'&'#197#203',..',
83: D 2 '0..+123-456*789/E()^1234567890''#171#181''#201#179',..'],
84: D 2 { german }
85: D 2 t1[0..+123-456*789/E()^1234567890''#222''#207'+''#206#204',..',
86: D 2 '0..+123-456*789/E()^1234567890''#219''#218#216',..'],
87: D 2 { swedish/finnish }
88: D 2 t1[0..+123-456*789/E()^1234567890''#197#212#207#206#204',..',
89: D 2 '0..+123-456*789/E()^1234567890''#220#208#219#218#216',..'],
90: D 2 { spanish }
91: D 2 t1[0..+123-456*789/E()^1234567890''#168#179''#183',..',
92: D 2 '0..+123-456*789/E()^1234567890''#185''#184'()=?/{}'#182'@;..'],
93: D 2 { katakana }
94: D 2 t1[
95: D 2 '0..+123-456*789/E()^1234567890''#199#204#177#179#180#181#212#213#214#220#206#205#208#219#218#185#200#217#210,
96: D 2 '0..+123-456*789/E()^1234567890''#199#204#167#169#170#171#172#173#174#166#178#205#222#223#218#185#164#161#165]
97: D 2 ]; { KATAKANA TABLE IS NIMITZ BASED AS IN 2.x - MODS LATER IN CODE }
98: D 2 { end of keylookup }
99: D 2 type
100: D 2 k2 = packed array [100..125] of 0..255;
101: D 2 bytealphabettype = packed array [boolean] of k2;
102: S
103: D 2 const
104: D 2 yencode=92;
105: D 2 kanaalphabet = bytealphabettype
106: D 2 [k2[ 215,190,201,216,192,195,178,189,182,221,197,198,
107: D 2 193,196,188,202,183,184,207,211,194,187,191,203,
108: D 2 186,208 ]
109: D 2 ,k2[ 215,190,201,216,192,195,168,189,182,221,197,198,
110: D 2 193,196,188,202,183,184,207,211,175,187,191,203,
111: D 2 186,208 ]];
112: S
113: D -2 var
114: D 1: -32768..32767;
115: D 1 begin
116: D 1 if (kbdlang=katakana_kbd) or
117: D 1 { (kbdtype=largekbd) or (kbdtype=smallkbd) } and
118: D 1 { (kbdlang=french_kbd) and (kbdlang=spanish_kbd) } and
119: D 1 (langtable[0]<>nil)
120: D 1 then
121: D 1 if (langtable[0]^langcode=us_kbd) then
```



```

121:C      4      begin
122:C      4      initlang := true;
123:C      4      with langtable[0]^ do
124:C      5      begin
125:C      5          { extend char keys don't exist on non ITF keyboards }
126:C      5          { and katakana wants pass thru mode }
127:C      5          transmode := kpass_extc;
128:C      5          langcode := kbdlang;
129:C      5
130:C      5          if kbdlang=katakana_kbd then      { 5/14/84 RQ/SFB }
131:C      6          begin
132:C      6              if langtable[1]=nil then new(langtable[1]);
133:C      6              langtable[1]:=langtable[0]^; { copy the us table }
134:C      6          end;
135:C      5
136:C      5          for i:=60 to 98 do      { MOVED 5/14/84 RQ/SFB }
137:C      6          begin { special alpha loading }
138:C      6              keytable[i].keys[false]:=keylookup[kbdlang,false,i];
139:C      6              keytable[i].keys[true]:=keylookup[kbdlang,true,i];
140:C      6          end;
141:C      5          keytable[99].keys[false]:=' '; keytable[99].keys[true]:=' ';
142:C      5
143:C      5          if kbdlang=katakana_kbd then
144:C      6          begin
145:C      6              semantics := katakanastrans;
146:C      6              can_nonadv:= false;
147:C      6              for i:=100 to 125 do
148:C      7              begin { load alpha keys }
149:C      7                  keytable[i].keys[false]:=chr(kanaalphabet[false,i]);
150:C      7                  keytable[i].keys[true]:=chr(kanaalphabet[true,i]);
151:C      7              end;
152:C      6              keytable[24].keyclass:=ignored_key; { ignore capslock key }
153:C      6
154:C      6              if kbdtype=ITFkbd then
155:C      7              begin { changes to keys 1, 2, 92 and 93 }
156:C      7                  keytable[1].keys[false]:=#219;
157:C      7                  keytable[1].keys[true]:=#176;
158:C      7                  keytable[2].keys[false]:=#209;
159:C      7                  keytable[2].keys[true]:=#163;
160:C      7                  keytable[92].keys[false]:=#222;
161:C      7                  keytable[92].keys[true]:=#222;
162:C      7                  keytable[93].keys[false]:=#223;
163:C      7                  keytable[93].keys[true]:=#162;
164:C      7              end;
165:C      6              IF (KBDTYPE = SMALLKBD) OR (KBDTYPE = LARGEKBD) THEN { RQ/SFB 5/11/84 }
166:C      7              WITH LANGTABLE[1]^ DO
167:C      8              BEGIN
168:C      8                  KEYTABLE[92].KEYS[TRUE] := '\';
169:C      8                  KEYTABLE[93].KEYS[TRUE] := '|';
170:C      8              END;
171:C      6              for i:=0 to 127 do keytable[i].no_capslock:=true;
172:C      6          end;
173:C      5
174:C      5
175:C      5      case kbdlang of
176:C      6      german_kbd,swedish_kbd:
177:C      6          begin
178:C      6              keytable[92].no_capslock:=false;
179:C      6              keytable[94].no_capslock:=false;
180:C      6              keytable[95].no_capslock:=false;

```

```

181:C      6          if kbdlang=german_kbd then
182:C      7          begin
183:C      7              keytable[120]:=keytable[109]; {z<=>y}
184:C      7              keytable[109].keys[false]:='z'; keytable[109].keys[true]:='Z';
185:C      7          end
186:C      7          else
187:C      7          begin { swedish_kbd }
188:C      7              keytable[91].no_capslock:=false;
189:C      7              keytable[93].no_capslock:=false;
190:C      7          end;
191:C      6          end;
192:C      6          spanish_kbd: BEGIN      { 5/10/84 SFB }
193:C      6              keytable[94].no_capslock:=false;
194:C      6              KEYTABLE[91].KEYCLASS:=NONA_ALPHA_KEY; { 5/11/84 SF8 }
195:C      6          END;
196:C      6          FRENCH_K8D : KEYTABLE[91].KEYCLASS:=NONADV_KEY; { 5/11/84 SF8 }
197:C      6          otherwise
198:C      6          end; { case }
199:C      5          langindex:=0;
200:C      5          kbdcapslock:=true; kbdaltlock:=(kbdlang=katakana_kbd);
201:C      5          end; { with langtable[0]^ }
202:C      5
203:C      4      end { if my kind of keyboard }
204:C      4      else initlang:=false;
205:C      2      end; { initlang }
206:C      5
207:C      1      end; { module nonitflang }
208:C      5
209:C      1      import NON_US_K8D1, loader;
210:C      5
211:C      1      begin
212:C      1          if initlang then markuser;
213:C      1          end.
214:C      5

```

No errors. No warnings.

***** Nonstandard language features enabled *****

NONUSKBD2

Description

NONUSKBD2 provides initialization for handling HPHIL non-U.S.-language keyboards.

Requirements

SYSDEVS, MISC, SYSGLOBALS, and ASM.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```
1:0 (*
2:0
3:0
4:0 (c) Copyright Hewlett-Packard Company, 1984.
5:0 All rights are reserved. Copying or other
6:0 reproduction of this program except for archival
7:0 purposes is prohibited without the prior
8:0 written consent of Hewlett-Packard Company.
9:0
10:0
11:0
12:0 RESTRICTED RIGHTS LEGEND
13:0 Use, duplication, or disclosure by the Government
14:0 is subject to restrictions as set forth in
15:0 paragraph (b) (3) (B) of the Rights in Technical
16:0 Data and Computer Software clause in
17:0 DAR 7-104.9(a).
18:0
19:0 HEWLETT-PACKARD COMPANY
20:0 0 Fort Collins, Colorado *)
21:0
22:0 0 $SYSPROGS
23:0 0 $heap dispose off$
24:0 0 $iocheck off$
25:0 0 $range off$ $ovfcheck off$
26:0 0 $debug off$
27:0 0 $stackcheck off$
28:0
29:0 0 program NON_US_KBD2INIT;
30:0 1 import sysdevs,misc;
31:0
32:0 1 type
33:0 1 t1 = packed array[boolean, 80..98] of char;
34:0 1 t2 = packed array[finish_kbd..swedish_kbd] of t1;
35:0 1 t3 = packed array[boolean, 1..2] of char;
36:0 1 t4 = packed array[finish_kbd..swedish_kbd] of t3;
37:0
38:0 1 const
39:0 1 lowkeys = t4[
40:0 1 {finish} t3['<','>'],
41:0 1 {belgian} t3['$<','>'],
42:0 1 {canadian eng} t3['@','#'],
43:0 1 {canadian fr} t3['@','#'],
44:0 1 {norwegian} t3['<','>'],
45:0 1 {danish} t3['<','>'],
46:0 1 {dutch} t3['@','>'],
47:0 1 {swiss gr} t3['$','3'],
48:0 1 {swiss fr} t3['$','3'],
49:0 1 {spanish_eur} t3['<','>'],
50:0 1 {spanish_lat} t3['@','>'],
51:0 1 { ABOVE [FALSE,1] CHANGED TO '5/8/84 SFB }
52:0 1 { ABOVE [TRUE,1] CHANGED TO '5/8/84 SFB }
53:0 1 { ABOVE [FALSE,2] CHANGED TO '5/10/84 SFB }
54:0 1 {uk} t3['<','>'],
55:0 1 {italian} t3['<','>'],
56:0 1 {french} t3['$<','>'],
57:0 1 {german} t3['<','>'],
58:0 1 {swedish} t3['<','>'],
59:0 1
60:0 1 specials = t2[
```

```
61:0 1 {finish} t1['1234567890*E0NL;-'],
62:0 1 {belgian} t1['1234567890*E0NL;-'],
63:0 1 {canadian eng} t1['1234567890*E0NL;-'],
64:0 1 {canadian fr} t1['1234567890*E0NL;-'],
65:0 1 {danish} t1['1234567890*E0NL;-'],
66:0 1 {dutch} t1['1234567890*E0NL;-'],
67:0 1 {french} t1['1234567890*E0NL;-'],
68:0 1 {german} t1['1234567890*E0NL;-'],
69:0 1 {italian} t1['1234567890*E0NL;-'],
70:0 1 {norwegian} t1['1234567890*E0NL;-'],
71:0 1 {spanish_eur} t1['1234567890*E0NL;-'],
72:0 1 {spanish_lat} t1['1234567890*E0NL;-'],
73:0 1 {swiss gr} t1['1234567890*E0NL;-'],
74:0 1 {swiss fr} t1['1234567890*E0NL;-'],
75:0 1 {uk} t1['1234567890*E0NL;-'],
76:0 1 { ABOVE [FALSE,95] CHANGED TO '5/8/84 SFB }
77:0 1 { ABOVE [TRUE,95] CHANGED TO '5/8/84 SFB }
78:0 1 { ABOVE [FALSE,91] CHANGED TO '5/8/84 SFB }
79:0 1 { ABOVE [TRUE,91] CHANGED TO '5/8/84 SFB }
80:0 1 { ABOVE [FALSE,93] CHANGED TO '5/8/84 SFB }
81:0 1 { ABOVE [TRUE,93] CHANGED TO '5/8/84 SFB }
82:0 1 { ABOVE [FALSE,91] CHANGED TO '5/8/84 SFB }
83:0 1 { ABOVE [TRUE,91] CHANGED TO '5/8/84 SFB }
84:0 1 { ABOVE [FALSE,95] CHANGED TO '5/8/84 SFB }
85:0 1 { ABOVE [TRUE,95] CHANGED TO '5/8/84 SFB }
86:0 1 { ABOVE [FALSE,91] CHANGED TO '5/8/84 SFB }
87:0 1 { ABOVE [TRUE,91] CHANGED TO '5/8/84 SFB }
88:0 1 { ABOVE [FALSE,93] CHANGED TO '5/8/84 SFB }
89:0 1 { ABOVE [TRUE,93] CHANGED TO '5/8/84 SFB }
90:0 1 { ABOVE [FALSE,91] CHANGED TO '5/8/84 SFB }
91:0 1 { ABOVE [TRUE,91] CHANGED TO '5/8/84 SFB }
92:0 1 { ABOVE [FALSE,95] CHANGED TO '5/8/84 SFB }
93:0 1 { ABOVE [TRUE,95] CHANGED TO '5/8/84 SFB }
94:0 1 { ABOVE [FALSE,91] CHANGED TO '5/8/84 SFB }
95:0 1 { ABOVE [TRUE,91] CHANGED TO '5/8/84 SFB }
96:0 1 { ABOVE [FALSE,93] CHANGED TO '5/8/84 SFB }
97:0 1 { ABOVE [TRUE,93] CHANGED TO '5/8/84 SFB }
98:0 1 { ABOVE [FALSE,91] CHANGED TO '5/8/84 SFB }
99:0 1 { ABOVE [TRUE,91] CHANGED TO '5/8/84 SFB }
100:0 1 { ABOVE [FALSE,95] CHANGED TO '5/8/84 SFB }
101:0 1 { ABOVE [TRUE,95] CHANGED TO '5/8/84 SFB }
102:0 1 { ABOVE [FALSE,91] CHANGED TO '5/8/84 SFB }
103:0 1 { ABOVE [TRUE,91] CHANGED TO '5/8/84 SFB }
104:0 1 { ABOVE [FALSE,93] CHANGED TO '5/8/84 SFB }
105:0 1 { ABOVE [TRUE,93] CHANGED TO '5/8/84 SFB }
106:0 1 { ABOVE [FALSE,91] CHANGED TO '5/8/84 SFB }
107:0 1 { ABOVE [TRUE,91] CHANGED TO '5/8/84 SFB }
108:0 1 { ABOVE [FALSE,95] CHANGED TO '5/8/84 SFB }
109:0 1 { ABOVE [TRUE,95] CHANGED TO '5/8/84 SFB }
110:0 1 { ABOVE [FALSE,91] CHANGED TO '5/8/84 SFB }
111:0 1 { ABOVE [TRUE,91] CHANGED TO '5/8/84 SFB }
112:0 1 { ABOVE [FALSE,93] CHANGED TO '5/8/84 SFB }
113:0 1 { ABOVE [TRUE,93] CHANGED TO '5/8/84 SFB }
114:0 1 { ABOVE [FALSE,91] CHANGED TO '5/8/84 SFB }
115:0 1 { ABOVE [TRUE,91] CHANGED TO '5/8/84 SFB }
116:0 1 { ABOVE [FALSE,95] CHANGED TO '5/8/84 SFB }
117:0 1 { ABOVE [TRUE,95] CHANGED TO '5/8/84 SFB }
118:0 1 { ABOVE [FALSE,91] CHANGED TO '5/8/84 SFB }
119:0 1 { ABOVE [TRUE,91] CHANGED TO '5/8/84 SFB }
120:0 1 { ABOVE [FALSE,93] CHANGED TO '5/8/84 SFB }
121:0 1 { ABOVE [TRUE,93] CHANGED TO '5/8/84 SFB }
```



```
121:D      1      ];
122:D
123:D      1 procedure doinit;
124:D      2 var
125:D      -2      i : -32768..32767;
126:C      2 begin
127:C      2   if (kbdtype=ITFkbd) and
128:C      3   (kdblang=>finnish_kbd) and (kdblang<>swedish_kbd) and
129:C      3   (langtable[0]<>nill)
130:C      3 then
131:C      3   if (langtable[0]^langcode=us_kbd) then
132:C      4   begin
133:C      4     with langtable[0]^do
134:C      5     begin
135:C      5       { extend char keys are extension bit }
136:C      5       transmode:=kshift_extc;
137:C      5       langcode:=kdblang;
138:C      5       { load special alpha keys }
139:C      5       keytable[1].keys[false]:=lowkeys[kdblang,false,1];
140:C      5       keytable[1].keys[true]:=lowkeys[kdblang,true,1];
141:C      5       keytable[2].keys[false]:=lowkeys[kdblang,false,2];
142:C      5       keytable[2].keys[true]:=lowkeys[kdblang,true,2];
143:C
144:C      5       for i:=80 to 98 do
145:C      6       with keytable[i] do
146:C      7       begin
147:C      7         keys[false]:=specials[kdblang,false,i];
148:C      7         keys[true]:=specials[kdblang,true,i];
149:C      7       end;
150:C
151:C      5   case kdblang of { language fixups }
152:C      6     finish_kbd,SWEDISH_KBD: { ADDED SWEDISH 5/10/84 SFB }
153:C      6     for i:=91 to 95 do keytable[i].no_capslock:=false;
154:C      6     { CHANGED BOTTOM INDEX OF ABOVE LOOP TO 91 5/8/84 SFB }
155:C      6     italian_kbd,french_kbd,belgian_kbd:
156:C      6     begin
157:C      6       keytable[119].no_capslock:=true;
158:C      6       keytable[94].no_capslock:=false;
159:C      6       { exchange z-w and m->? }
160:C      6       keytable[119].keys[false]:='.'; keytable[119].keys[true]:='?';
161:C      6       keytable[105]:=keytable[120]; { z<-w }
162:C      6       keytable[120].keys[false]:='w'; keytable[120].keys[true]:='W';
163:C      6       { ABOVE CHANGED TO 'w' 'W' 5/8/84 SFB }
164:C      6       if kdblang<>italian_kbd then
165:C      7       begin { exchange a-q }
166:C      7         keytable[104]:=keytable[112]; { q<-a }
167:C      7         keytable[112].keys[false]:='q'; keytable[112].keys[true]:='Q';
168:C      7         { ABOVE CHANGED TO [112].KEYS[TRUE] 5/8/84 SFB }
169:C      7       end
170:C      7       ELSE
171:C      7         KEYTABLE[87].KEYCLASS:=NONA_ALPHA_KEY; { 5/9/84 SFB }
172:C      6     end;
173:C      6     cdn_eng_kbd,cdn_fr_kbd:
174:C      6     begin
175:C      6       keytable[93].no_capslock:=false;
176:C      6       keytable[98].no_capslock:=false;
177:C      6       KEYTABLE[95].KEYCLASS:=NONA_ALPHA_KEY; { 5/9/84 SFB }
178:C      6     end;
179:C      6     norwegian_kbd,german_kbd,danish_kbd: { REMOVED SWEDISH 5/10/84 SFB }
180:C      6     begin
```

```
181:C      6     keytable[92].no_capslock:=false;
182:C      6     keytable[94].no_capslock:=false;
183:C      6     keytable[95].no_capslock:=false;
184:C      6     if kdblang=german_kbd then
185:C      7     begin { switch z-y }
186:C      7       keytable[120]:=keytable[109]; { y<-z }
187:C      7       keytable[109].keys[false]:='z'; keytable[109].keys[true]:='Z';
188:C      7       { ABOVE CHANGED TO [109].KEYS[TRUE] 5/8/84 SFB }
189:C      7     end;
190:C      6     end;
191:C      6     swiss_gr_kbd,swiss_fr_kbd:
192:C      6     begin { switch z-y }
193:C      6       keytable[120]:=keytable[109]; { y<-z }
194:C      6       keytable[109].keys[false]:='z'; keytable[109].keys[true]:='Z';
195:C      6       { ABOVE CHANGED TO [109].KEYS[TRUE] 5/8/84 SFB }
196:C      6       KEYTABLE[93].KEYCLASS:=NONADV_KEY;
197:C      6     end;
198:C      6     uk_kbd,dutch_kbd; { no changes }
199:C      6     spanish_latn_kbd,spanish_eur_kbd: keytable[94].no_capslock:=false;
200:C      6     spanish_kbd,katakana_kbd; { not implemented here }
201:C      6     otherwise
202:C      6     end;
203:C      5     langindex:=0;
204:C      5     kbdcapslock:=true; kbdcaltlock:=false;
205:C      5     IF KBDLANG IN { THESE MODS 5/9/84 SFB }
206:C      6     [NORWEGIAN_KBD,DANISH_KBD,SWISS_GR_KBD,SWISS_FR_KBD] THEN
207:C      6     KEYTABLE[91].KEYCLASS:=NONADV_KEY;
208:C      5     IF KBDLANG IN
209:C      6     [NORWEGIAN_KBD,DANISH_KBD] THEN
210:C      6     KEYTABLE[93].KEYCLASS:=NONA_ALPHA_KEY;
211:C      5     IF KBDLANG IN
212:C      6     [DUTCH_KBD,SPANISH_EUR_KBD] THEN
213:C      6     KEYTABLE[95].KEYCLASS:=NONADV_KEY;
214:C      5     IF KBDLANG = DUTCH_KBD THEN { BUG FIX 9/20/84 SFB }
215:C      6     KEYTABLE[92].KEYCLASS:=NONA_ALPHA_KEY;
216:C      5     IF KBDLANG = FRENCH_KBD THEN { BUG FIX 9/20/84 SFB }
217:C      6     KEYTABLE[92].KEYCLASS:=NONADV_KEY;
218:C      5     IF KBDLANG IN
219:C      6     [BELGIAN_KBD,SPANISH_LATIN_KBD,CDN_ENG_KBD,CDN_FR_KBD] THEN
220:C      6     KEYTABLE[92].KEYCLASS:=NONADV_KEY;
221:C      5     IF KBDLANG=SPANISH_EUR_KBD THEN
222:C      6     KEYTABLE[91].KEYCLASS:=NONA_ALPHA_KEY;
223:C      5     end; { with langtable[0]^ }
224:C      4     end;
225:C      2 end; { doinit }
226:C      1 begin
227:C      1   doinit; { structure is to avoid global vars }
228:C      1 end.
```

No errors. No warnings.

***** Nonstandard language features enabled *****

PRINTER

Description

PRINTER is the transfer method (TM) for the system printer.

Requirements

IODECLARATIONS and whatever IOLIBRARY interface card driver that is applicable.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*)
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1984.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S     RESTRICTED RIGHTS LEGEND
11:S
12:S     Use, duplication, or disclosure by the Government
13:S     is subject to restrictions as set forth in
14:S     paragraph (b) (3) (8) of the Rights in Technical
15:S     Data and Computer Software clause in
16:S     DARF 7-104.3(a).
17:S
18:S     HEWLETT-PACKARD COMPANY
19:D     0 Fort Collins, Colorado      *)
20:S
21:S
22:D     0 $modcal$
23:D     0 $debug off, range off, ovflcheck off$
24:D     0 $stackcheck off, ioccheck off$
25:S
26:D     0 $search 'IOLIB:KERNEL', 'OSFS:SYSDEVS'$
27:S

```

```

28:D     0 $page$
29:D     0 $copyright 'COPYRIGHT (C) 1984 BY HEWLETT-PACKARD COMPANY'$
30:S
31:S
32:D     0 module prtdvr;
33:S
34:D     1 import
35:D     1   sysglobals,
36:D     1   iodeclarations,
37:D     1   asm, sysdevs, mini, misc, fs;
38:S
39:D     1 export
40:D     1   procedure prtio (fp: fibp; request: amrequesttype; anyvar buffer: window;
41:D     2   length, position: integer);
42:D     1 implement {prtdvr}
43:S
44:D     1 procedure bep;
45:D     2   begin write(bellchar); end;
46:S
47:D     1 procedure prtio(fp: fibp; request: amrequesttype; anyvar buffer: window;
48:D     2   length, position: integer);
49:S
50:D     2 const
51:D     2   uclr_timeout_const = 25;      {HPiB commands during unitclear}
52:S
53:D     2   repeating_timeout = 333;      {timeout constant after initial timeout}
54:D     2   timeouts_per_beep = 40;      {beep period in repeating timeout units}
55:S
56:D     2   SDC = 4;      {selective device clear}
57:D     2   TAGbase = 32;      {listen address group base}
58:D     2   TAGbase = 64;      {talk address group base}
59:S
60:D     2   linefeed = chr(10);      {ASCII linefeed}
61:D     2   formfeed = chr(12);      {ASCII formfeed}
62:D     2   return = chr(13);      {ASCII carriage return}
63:S
64:D     2 var
65:S
66:D     -2 2   select_code: type_isc;
67:D     -6 2   sc_table_entry_ptr: ^isc_table_type;
68:D     -10 2  previous_char_ptr: charptr;
69:D     -12 2  bus_address: byte;
70:S
71:D     -13 2 channel_is_setup: boolean;
72:D     -14 2 writing_previous_char: boolean;
73:D     -15 2 previously_timed_out: boolean;
74:D     -16 2 timeout_blanked: boolean;
75:D     -20 2 user_spec_timeout: integer;
76:D     -24 2 current_timeout: integer;
77:D     -26 2 timeout_counter: shortint;
78:D     -70 2 saved_line : string[42];      { 3.0 bug fix -- 4/12/84 }
79:D     -71 2 line_needs_restoring: boolean; { 4/12/84 }
80:D     -76 2 buf: charptr;
81:D     -77 2 saved_echo: boolean;      { 5/9/84 }
82:S

```



```

83:D -77 2 $page$
84:S
85:D 2 procedure reset_card_and_confirm_timeout;
86:D 3 var
87:D -2 3 saved_escapecode: shortint;
88:D -6 3 saved_ioe_sc: integer;
89:D -10 3 saved_ioe_result: integer;
90:C 3 begin (reset_card_and_confirm_timeout)
91:C 3 saved_escapecode := _escapecode;
92:C 3 saved_ioe_sc := ioe_isc;
93:C 3 saved_ioe_result := ioe_result;
94:C 3 try
95:C 4 with sc_table_entry_ptr^ do
96:C 5 call(io_drv_ptr^.iod_init, io_tmp_ptr);
97:C 4 recover
98:C 4 if (escapecode<>ioescapecode) or (ioe_isc<>select_code) then
99:C 5 escape(escapecode);
100:C 3 ioe_isc := saved_ioe_sc;
101:C 3 ioe_result := saved_ioe_result;
102:C 3 if (saved_escapecode<>ioescapecode) or (ioe_isc<>select_code) then
103:C 4 escape(saved_escapecode);
104:C 3 if ioe_result<>ioe_timeout then
105:C 4 ioresc(znodevice);
106:C 3 end; (reset_card_and_confirm_timeout)
107:S
108:D 2 procedure clear_unit;
109:D 3 procedure HPIBsd;
110:C 4 begin (HPIBsd)
111:C 5 with sc_table_entry_ptr^, io_drv_ptr^, io_tmp_ptr^ do
112:C 6 begin
113:C 7 call(iod_send, io_tmp_ptr, '?');
114:C 7 timeout := ucir_timeout_const;
115:C 7 call(iod_send, io_tmp_ptr, chr(LAGbase+bus_address));
116:C 7 call(iod_send, io_tmp_ptr, chr(SDC));
117:C 7 end; (with)
118:C 4 end; (HPIBsd)
119:C 3 begin (clear_unit)
120:C 4 with sc_table_entry_ptr^ do
121:C 5 if card_type=hpib_card then
122:C 6 try
123:C 7 HPIBsd; (first attempt)
124:C 6 recover
125:C 6 begin
126:C 7 reset_card_and_confirm_timeout;
127:C 7 try
128:C 8 HPIBsd; (second attempt)
129:C 7 recover
130:C 7 begin
131:C 8 reset_card_and_confirm_timeout;
132:C 8 ioresc(ztimeout);
133:C 8 end; (recover)
134:C 6 end (recover)
135:C 5 else
136:C 6 try
137:C 7 call(io_drv_ptr^.iod_init, io_tmp_ptr);
138:C 6 recover
139:C 6 if (escapecode<>ioescapecode) or (ioe_isc<>select_code) then escape(escapecode)
140:C 7 else ioresc(znodevice);
141:C 3 end; (clear_unit)
142:S

```

```

143:D -77 2 $page$
144:S
145:D 2 procedure wrtchar(character: char; last_char: boolean);
146:S
147:D 3 var
148:D -1 3 hs_successfully_initiated: boolean;
149:D -2 3 previous_hs_completed : boolean;
150:S
151:S
152:D 3 procedure restore_line;
153:D -1 4 var dummyc:char;
154:C 4 begin
155:C 5 if line_needs_restoring then ( 4/12/84 )
156:C 6 begin
157:C 7 keybuffer^.echo:=saved_echo;
158:C 7 keybufops(kdisplay,dummyc);
159:C 7 line_needs_restoring:=false;
160:C 5 end;
161:C 4 end;
162:S

```



```

163:D -2 3 $pages$
164:S
165:D 3 procedure inform_operator;
166:D -44 4 var lmsr : string[42]; ( 3.0 bug fix -- 4/12/84 )
167:C 4 begin {inform_operator}
168:S
169:C 4 if not previously_timed_out then
170:C 5 begin
171:C 5 timeout_blanked := true;
172:C 5 timeout_counter := 0;
173:C 5 end;
174:C 4 if not line_needs_restoring then
175:C 5 begin
176:C 5 saved_line := '* Printer timeout: fix or ';
177:C 5 if intlevel=0 then saved_line:=saved_line+ '<stop> aborts *';
178:C 6 else saved_line:=saved_line+'wait auto-abort*';
179:C 5 ( 3.0 bug fix -- 4/12/84 )
180:C 5 line_needs_restoring := true;
181:C 5 if menustate=m_none then saved_echo:=keybuffer^.echo
182:C 6 else saved_echo:=true;
183:C 5 menustate := m_none; ( 4/12/84 )
184:C 5 keybuffer^.echo :=false;
185:C 5 end;
186:C 4 if timeout_blanked then lmsr:= saved_line
187:C 5 else lmsr:= ' ';
188:C 4 CALL(CRLLHOOK,CLLDISPLAY,LMSR,' ');
189:C 4 timeout_blanked:= (timeout_counter mod 4)<>0;
190:S
191:C 4 if timeout_counter<=1 then bep;
192:C 4 timeout_counter := timeout_counter+1;
193:C 4 if timeout_counter>=timeout_per_beep then
194:C 5 if intlevel=0 then timeout_counter := 0
195:C 6 else
196:C 6 begin
197:C 6 bep;
198:C 6 restore_line;
199:C 6 ioresc(ztimeout);
200:C 6 end; {else}
201:S
202:C 4 end; {inform_operator}
203:S

```

```

204:D -2 3 $pages$
205:S
206:C 3 begin {wrtchar}
207:C 3 try
208:C 4 with sc_table_entry_ptr^, io_drv_ptr^, io_tmp_ptr^ do
209:C 4 repeat
210:C 5 try
211:C 7 previous_hs_completed := false;
212:C 7 if not channel_is_setup then
213:C 8 begin
214:C 8 case card_type of
215:C 9 hpib_card:
216:C 9 begin
217:C 9 call(iod_send, io_tmp_ptr, '?');
218:C 9 previous_hs_completed := true;
219:C 9 timeout := current_timeout;
220:C 9 call(iod_send, io_tmp_ptr, chr(TAGbase+addressed));
221:C 9 call(iod_send, io_tmp_ptr, chr(LAGbase+bus_address));
222:C 9 end; {hpib_card}
223:C 9 serial_card:
224:C 9 if card_id=hp98626 then {always set full duplex modem HS}
225:C 10 call(iod_wtc, io_tmp_ptr, 13, 1);
226:C 9 otherwise
227:C 9 {do nothing};
228:C 9 end; {case}
229:C 8 channel_is_setup := true;
230:C 8 end; {if}
231:C 7 call(iod_wtb, io_tmp_ptr, character);
232:C 7 previous_char_ptr^ := character;
233:C 7 timeout := current_timeout;
234:C 7 if last_char then
235:C 8 if card_type=hpib_card then
236:C 9 call(iod_send, io_tmp_ptr, '?');
237:C 8 if previously_timed_out then
238:C 8 if not writing_previous_char then
239:C 9 begin
240:C 9 restore_line;
241:C 9 current_timeout := user_spec_timeout;
242:C 9 previously_timed_out := false;
243:C 9 end; {if}
244:C 7 hs_successfully_initiated := true;
245:C 7 recover
246:C 7 begin
247:C 7 reset_card_and_confirm_timeout;
248:C 7 channel_is_setup := false;
249:C 7 inform_operator;
250:C 7 previously_timed_out := true;
251:C 7 current_timeout := repeating_timeout;
252:C 7 if not (writing_previous_char or previous_hs_completed) then
253:C 8 begin
254:C 8 writing_previous_char := true;
255:C 8 wrtchar(previous_char_ptr^, false);
256:C 8 writing_previous_char := false;
257:C 8 end; {if}
258:C 7 hs_successfully_initiated := false;
259:C 7 end; {recover}
260:C 6 until hs_successfully_initiated;
261:C 4 recover
262:C 4 begin
263:C 4 restore_line; ( 4/12/84 )

```



```

264:C      4      escape(escapecode);
265:C      4      end; {recover}
266:C      3      end; {wrtchar}

```

```

267:D      -77  2 $page$
268:S
269:C      2      begin {prtio}
270:C      2      ioresult := ord(ioerror);                      { scs 1/17/83 }
271:C      2      with unitable^[fp^.funit] do
272:C      3      begin
273:C      3      select_code := sc;
274:C      3      sc_table_entry_ptr := addr(isc_table[select_code]);
275:C      3      bus_address := ba;
276:C      3      previous_char_ptr := addr(dvrtemp);
277:C      3      user_spec_timeout := devid; {user-specified in CTABLE}
278:C      3      end; {with}
279:S
280:C      2      buf := addr(buffer);
281:C      2      channel_is_setup := false;
282:C      2      current_timeout := user_spec_timeout;
283:C      2      previously_timed_out := false;
284:C      2      writing_previous_char := false;
285:C      2      line_needs_restoring := false; { 4/12/84 }
286:S
287:C      2      try
288:C      3      with sc_table_entry_ptr^, io_tmp_ptr^ do
289:C      4      begin
290:C      4      if card_type=no_card then ioresc(znodevice);
291:C      4      while (in_bufptr<>nil) or (out_bufptr<>nil) do {nothing};
292:C      4      end; {with}
293:S
294:C      3      case request of
295:C      4      flush:
296:C      4      (do nothing);
297:C      4      clearunit:
298:C      4      clear unit;
299:C      4      writeeoI:
300:C      4      begin
301:C      4      wrtchar(return, false);
302:C      4      wrtchar(linefeed, true)
303:C      4      end;
304:C      4      writebytes:
305:C      4      while length>0 do
306:C      5      begin
307:C      5      wrtchar(buf^, length=1);
308:C      5      buf := addr(buf^, 1);
309:C      5      length := length-1;
310:C      5      end;
311:C      4      otherwise
312:C      4      ioresc(zbadmode);
313:C      4      end; {case}
314:C      3      recover
315:C      3      if (escapecode=-20) and previously_timed_out then
316:C      4      ioresult := ord(ztimeout)
317:C      4      else if escapecode<>-10 then
318:C      5      escape(escapecode);
319:S
320:C      2      end; {prtio}
321:S
322:C      1      end. {prtdvr}
323:S

```


RS_DRV

Description

RS_DRV contains low-level I/O drivers.

Usage

RS_DRV is used for the 98626 interface (and the Model 216/217's built-in RS-232 interface).

Requirements

RS and the I/O library kernel (IODECLARATIONS, etc.)

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1984.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S     RESTRICTED RIGHTS LEGEND
11:S
12:S     Use, duplication, or disclosure by the Government
13:S     is subject to restrictions as set forth in
14:S     paragraph (b) (3) (B) of the Rights in Technical
15:S     Data and Computer Software clause in
16:S     DAR 7-104.9(a).
17:S
18:S     HEWLETT-PACKARD COMPANY
19:D     0 Fort Collins, Colorado      *)
20:S
21:S
22:S     0 $SEARCH 'IOLIB:KERNEL.CODE'$
23:S
24:D     0 $COPYRIGHT 'COPYRIGHT (C) 1984 BY HEWLETT-PACKARD COMPANY'$
25:D     0 $MODCAL ON$
26:D     0 $PARTIAL EVAL ON$
27:D     0 $STACKCHECK ON$
28:D     0 $RANGE OFF$
29:D     0 $DEBUG OFF$
30:D     0 $OVFLCHECK OFF$

```

```

31:D     0 $PAGE$
32:S
33:D     0 {*****}
34:D     0 {*
35:D     0 {*      NOT RELEASED      VERSION 3.0
36:D     0 {*
37:D     0 {*****}
38:D     0 {*
39:D     0 {*
40:D     0 {*      IOLIB      RS_DRIVERS
41:D     0 {*
42:D     0 {*
43:D     0 {*****}
44:D     0 {*
45:D     0 {*
46:D     0 {*      library      - IOLIB
47:D     0 {*      name        - RS_DRIVERS
48:D     0 {*      module(s)   - init_rs
49:D     0 {*                  - rs
50:D     0 {*
51:D     0 {*      date        - 6 August 1982
52:D     0 {*      update      - 5 March 1984
53:D     0 {*      release     - ?????????????
54:D     0 {*
55:D     0 {*      source      - IOLIB:RS_DRV.TEXT
56:D     0 {*      object      - IOLIB:RS_DRV.CODE
57:D     0 {*
58:D     0 {*****}

```



```
59:D 0 $PAGES
60:D 0 *****
61:D 0 *
62:D 0 *
63:D 0 *      BUG FIX HISTORY
64:D 0 *
65:D 0 *      jws      -----      io_init_rs      Find 98644 cards
66:D 0 *      3/5/84
67:D 0 *
68:D 0 *
69:D 0 *
70:D 0 *
71:D 0 *
72:D 0 *
73:D 0 *
74:D 0 *
75:D 0 *
76:D 0 *
77:D 0 *
78:D 0 *
79:D 0 *****
```

```
80:D 0 $PAGES
81:D 0 *****
82:D 0 *
83:D 0 *
84:D 0 *      This is the source code for an external procedures library
85:D 0 *      to be used for general purpose interfacing on the HP 9826.
86:D 0 *
87:D 0 *      The library consists of 3 primary sets of modules -
88:D 0 *
89:D 0 *          1.      KERNEL modules
90:D 0 *          2.      driver modules
91:D 0 *          3.      IOLIB modules
92:D 0 *
93:D 0 *      The KERNEL modules consist of the following modules -
94:D 0 *
95:D 0 *          1.      iodeclarations ( contains static r/w space )
96:D 0 *          2.      iocomasm
97:D 0 *          3.      general_0      ( initialization & low level
98:D 0 *                               routines like ioread/iowrite)
99:D 0 *
100:D 0 *      The KERNEL modules also have an executable program segment
101:D 0 *      that gets executed at the time it is loaded. This program
102:D 0 *      initializes the static read/write memory. This program also
103:D 0 *      allocates the temporary storage for any card that exists -
104:D 0 *      independent of whether there is or is not a driver for it.
105:D 0 *
106:D 0 *      The driver modules consist of the actual assembly or PASCAL
107:D 0 *      routines that deal with a specific interface card. There is
108:D 0 *      also an executable program segment for each driver module.
109:D 0 *      This program searches the select code table in the static r/w
110:D 0 *      initialized by the KERNEL general_0 module for all select codes
111:D 0 *      that have the right interface card ( HPIB drivers will search
112:D 0 *      for the 98624 interface ). This program will then set up the
113:D 0 *      driver tables to point to the correct drivers.
114:D 0 *
115:D 0 *      The rest of the IOLIB modules are high-level modules that are
116:D 0 *      used by an end user in his/her application program.
117:D 0 *
118:D 0 *      The KERNEL and some set of driver modules will exist in the
119:D 0 *      INITLIB file as object code ( not EXPORT text ). The
120:D 0 *      export text will reside on the IO file. The rest
121:D 0 *      of the library will reside on the IO file.
122:D 0 *****
```



```

123:D 0 $PAGE$
124:D 0 {*****}
125:D 0 {*}
126:D 0 {*}
127:D 0 {* REFERENCES :}
128:D 0 {*}
129:D 0 {*}
130:D 0 {* 1. 9826 I/O Designers Guide ( ----- )}
131:D 0 {*}
132:D 0 {* 2. 68000 Manual ( Motorola )}
133:D 0 {*}
134:D 0 {* 3. Pascal Language System Users Manual ( ----- )}
135:D 0 {*}
136:D 0 {* 4. Pascal Procedure Library Users Manual( ----- )}
137:D 0 {*}
138:D 0 {* 5. 9826 card documentation ( ----- )}
139:D 0 {*}
140:D 0 {* 6. Pascal I/O Library IRS ( ----- )}
141:D 0 {*}
142:D 0 {* 7. 98626A ERS ( ----- )}
143:D 0 {*}
144:D 0 {*}
145:D 0 {*****}
146:S 0
147:D 0 PROGRAM rs_initialize ( INPUT , OUTPUT );

```

```

148:D 1 $PAGE$
149:D 1 {*****}
150:D 1 {*}
151:D 1 {* RS-232 CARD DRIVERS}
152:D 1 {*}
153:D 1 {*}
154:D 1 {*}
155:D 1 {*****}
156:D 1 EXTERNAL MODULE rs;
157:S 1 ( update 2/14/83
158:S 1
159:S 1 purpose This module is a declaration of the importation text for
160:S 1 the external drivers.
161:S 1
162:S 1 note The assembly language code that is imported needs to be
163:S 1 called 'rs'. The routines need to be called
164:S 1 'rs_000000'
165:D 1 )
166:S 1
167:D 1 IMPORT sysglobals, iodeclarations ;
168:S 1
169:D 1 EXPORT
170:S 1
171:S 1
172:D 1 1 PROCEDURE rs_init ( temp : ANYPTR );
173:D 1 1 PROCEDURE rs_isr ( temp : PISRB );
174:D 1 1 PROCEDURE rs_rdb ( temp : ANYPTR ; VAR x : CHAR);
175:D 1 1 PROCEDURE rs_wtb ( temp : ANYPTR ; val : CHAR);
176:D 1 1 PROCEDURE rs_rdw ( temp : ANYPTR ; VAR x : io_word);
177:D 1 1 PROCEDURE rs_wtw ( temp : ANYPTR ; val : io_word);
178:D 1 1 PROCEDURE rs_rds ( temp : ANYPTR ; reg : io_word;
179:D 1 2 VAR x : io_word);
180:D 1 1 PROCEDURE rs_wtc ( temp : ANYPTR ; reg : io_word;
181:D 1 2 val : io_word );
182:D 1 1 PROCEDURE rs_tfr ( temp : ANYPTR ; bcb : ANYPTR );
183:S 1
184:S 1
185:S 1
186:S 1
187:D 1 END; { of rs }

```



```

188:D      1 $PAGES
189:D      1 MODULE init_rs;
190:S      ( update 3/5/84 jws
191:S      purpose This module initializes the RS 232 drivers.
192:S
193:S      1
194:D      1 )
195:S
196:S
197:D      1 IMPORT iodeclarations ;
198:S
199:D      1 EXPORT
200:S
201:S
202:D      1 VAR
203:D      -120 1 rs_drivers : drv_table_type;
204:S
205:D      -120 1 PROCEDURE io_init_rs;
206:S
207:S
208:S
209:D      -120 1 IMPLEMENT
210:S
211:S
212:D      -120 1 IMPORT sysglobals ,
213:D      -120 1 isr ,
214:D      -120 1 general_0 ,
215:D      -120 1 rs ;
216:S
217:D      1 PROCEDURE io_init_rs;
218:D      2 VAR
219:D      -2 2 io_isc : type_isc;
220:D      -4 2 dummyword : io_word;
221:D      -6 2 io_lvl : io_byte;
222:C      2 BEGIN
223:S
224:C      2 io_revid := io_revid + ' R3.0'; (vers number changed JS 8/3/83 )
225:S
226:C      2 ( set up the driver tables )
227:S
228:C      2 WITH rs_drivers DO BEGIN
229:C      3 rs_drivers := dummy_drivers ;
230:C      3 iod_init := rs_init;
231:C      3 iod_isr := rs_isr;
232:C      3 iod_rdb := rs_rdb;
233:C      3 iod_wtb := rs_wtb;
234:C      3 iod_rdw := rs_rdw;
235:C      3 iod_wtw := rs_wtw;
236:C      3 iod_rds := rs_rds;
237:C      3 iod_wtc := rs_wtc;
238:C      3 iod_tfr := rs_tfr;
239:C      3 END; ( of WITH )
240:S
241:S
242:C      2 ( set up drivers for the interfaces )
243:S
244:C      2 FOR io_isc:=iomisc TO iomaxisc DO
245:C      3 WITH isc_table[io_isc] DO BEGIN
246:S
247:C      4 IF (card_id = hp98626) or (card_id=hp98644) ( jws 3/5/84 )

```

```

248:C      5 THEN BEGIN
249:S
250:C      5 io_drv_ptr:=ADDR(rs_drivers);
251:S
252:C      5 ( if the card exists then link in an ISR for it )
253:C      5 ( ??? - what happens if an ISR fires during init )
254:S
255:C      5 io_lvl:=((io_read_byte(io_isc,3) DIV 16) MOD 4)+3;
256:C      5 IF io_tmp_ptr^.myisrib.INTERFACEADDR <> NIL
257:C      6 THEN BEGIN
258:C      6 ( if isr exists then unlink it )
259:C      6 ISRUNLINK(io_lvl,^.myisrib);
260:C      6 ADDR(io_tmp_ptr^.myisrib)); ( level
261:C      6 END; ( of IF ) ( isrib info )
262:S
263:C      5 PERMISSRLINK(io_drv_ptr^.iod_isr, ( isr
264:C      5 ANYPTR(INTEGER(card_ptr)+3), ( card address )
265:C      5 192, ( intr. mask )
266:C      5 192, ( intr. value )
267:C      5 io_lvl, ( level
268:C      5 ADDR(io_tmp_ptr^.myisrib)); ( isrib info )
269:C      5 END; ( of IF card_type = hp98626 )
270:S
271:C      4 END; ( of FOR io_isc WITH isc_table[io_isc] BEGIN )
272:S
273:S
274:C      2 ( call the actual driver initialization )
275:S
276:C      2 FOR io_isc:=iomisc TO iomaxisc DO
277:C      3 WITH isc_table[io_isc] DO
278:C      4 IF (card_id = hp98626) OR (card_id = hp98644) ( jws 3/5/84 )
279:C      5 THEN BEGIN
280:S
281:C      5 CALL( io_drv_ptr^.iod_init, io_tmp_ptr );
282:S
283:C      5 END; ( of WITH IF )
284:S
285:S
286:C      2 END; ( of io_init_rs )
287:S
288:C      1 END; ( of MODULE init_rs )

```



```
289:D      1 $PAGES
290:S
291:D      1 IMPORT    init_rs ,
292:D      1          LOADER ;
293:S
294:S
295:C      1 BEGIN
296:C      1   io init_rs;
297:C      1   MARKUSER;
298:C      1 END. { of rs_initialize }
```

No errors. No warnings.

***** Nonstandard language features enabled *****

SEGMENTER

Description

SEGMENTER provides for the loading, calling, and unloading of code segments.

Requirements

LOADER, SYSGLOBALS, ASM, LDR, MISC, and FS.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:D      0 $modcal$
2:S
3:D      0 { SEGMENTER module --
4:D      0 {
5:D      0 {
6:D      0 {
7:D      0 {
8:D      0 {
9:D      0 {
10:D     0 {
11:D     0 {
12:D     0 {
13:D     0 {
14:D     0 {
15:S
16:D
17:S      0 module segmenter;
18:D      1 import loader, ldr, sysglobals, MISC;
19:S
20:D      1 export
21:D      1   type segment_proc = procedure;
22:D      1   proc_name      = string[120];
23:S
24:D      1   procedure init_segmenter(anyvar lowcode, highcode,
25:D      2   lowglobal, highglobal: byte);
26:S
27:D      1   procedure load_segment (filename: fid);
28:D      1   procedure load_heap_segment(filename: fid);
29:D      1   procedure unload_segment;
30:D      1   procedure unload_all;
31:S
32:D      1   procedure call_segment (filename: fid);
33:D      1   procedure call_segment_proc(filename: fid; symbol: proc_name);
34:S
35:D      1   function find_proc(symbol: proc_name): segment_proc;
36:D      1   function exists_proc(p: segment_proc): boolean;
37:S
38:D      1   procedure segment_space(var code, global: integer);
39:S
40:D      1 implement
41:S
42:D      1 external module asm;
43:D      1 export
44:D      1   function allocate(size: integer): anyptr;
45:D      1   procedure newbytes(var p: anyptr; size: integer);
46:D      1 end;
47:S
48:D      1 import asm;
49:S
50:D      1 type proc = procedure;
51:S
52:D      1   trick_proc = record case boolean of
53:D      1   true: (p: proc);
54:D      1   false: (ep, sl: integer);
55:D      1   end;
56:D      1
57:D      1   state_ptr = ^state_rec;
58:D      1   name_ptr  = ^proc_name;
59:S
60:D      1   state_rec = record

```

```

61:D      1   savelist: state_ptr;
62:D      1   restore_heap: boolean;
63:D      1   saveheap: ^integer;
64:D      1   saveglob: ^integer;
65:D      1   savecod: integer;
66:D      1   saveentry: moddescptr;
67:D      1   savedef: moddescptr;
68:D      1   savefiles: anyptr;
69:D      1   end;
70:S
71:D      -8 1 var highglob, lowglob: integer;
72:D      -16 1 lowcod, highcod: integer;
73:D      -20 1 segment_list: state_ptr;
74:S
75:D      1 procedure init_segmenter(anyvar lowcode, highcode,
76:D      2 lowglobal, highglobal: byte);
77:C
78:C      2 begin
79:C      2   lowcod := integer(addr(lowcode));
80:C      2   highcod := integer(addr(highcode));
81:C      2   if highcod < lowcod then begin
82:C      3   lowcod := integer(addr(highcode));
83:C      3   highcod := integer(addr(lowcode));
84:C      3   end;
85:C      2   lowglob := integer(addr(lowglobal));
86:C      2   highglob := integer(addr(highglobal));
87:C      2   if highglob < lowglob then begin
88:C      3   lowglob := integer(addr(highglobal));
89:C      3   highglob := integer(addr(lowglobal));
90:C      3   end;
91:C      2   if odd(lowcod) then lowcod := lowcod + 1;
92:C      2   if odd(lowglob) then lowglob := lowglob + 1;
93:C      2   if odd(highcod) then highcod := highcod - 1;
94:C      2   if odd(highglob) then highglob := highglob - 1;
95:C      2 end;
96:D      1 procedure segment_space(var code, global: integer);
97:C
98:C      2 code := highcod - lowcod;
99:C      2 global := highglob - lowglob;
100:C      2 end;
101:S
102:C      1 procedure no_proc; begin escape(120); end;
103:S
104:D      1 function exists_proc(p: segment_proc): boolean;
105:C
106:C      2 begin
107:C      2   exists_proc := not(p = no_proc);
108:C      2 end;
109:S
110:D      -122 1 function find_proc(symbol: proc_name): segment_proc;
111:D      -122 2 var
112:D      -126 2   modp: moddescptr;
113:D      -134 2   ptr, valueptr: addrec;
114:D      -135 2   found: boolean;
115:D      -144 2   proc_rec: trick_proc;
116:C
117:C      2 begin
118:C      2   find_proc := no_proc;
119:C      2   found := false;
120:C      2   modp := sysdefs;
121:C      2   while (modp <> nil) and not found do
122:C      3   with modp do

```



```

121:C      4      begin
122:C      4      ptr := defaddr;
123:C      4      while (ptr.a < defaddr.a + defsize) and not found do
124:C      4      begin
125:C      5      found := ptr.syp = symbol;
126:C      5      ptr.a := ptr.a + strlen(ptr.syp) - 1;
127:C      5      ptr.a := ptr.a + ord(odd(ptr.a));
128:C      5      valueptr.a := ptr.a + 2;
129:C      5      if found then with proc_rec do
130:C      5      begin
131:C      6      sl := 0;
132:C      6      ep := valueptr.vcp.value;
133:C      6      find_proc := p;
134:C      6      end;
135:C      5      ptr.a := ptr.a + ptr.gvp.short;
136:C      5      end; {while}
137:C      4      modp := link;
138:C      4      end; {with modp}
139:C      2      end;
140:S
141:D      1      procedure save_state(var state: state_rec; heapoff, codeoff: integer);
142:C      1      begin with state do
143:C      1      begin
144:C      2      savelist := segment_list;      segment_list := addr(state);
145:C      2      mark(saveheap);      restore_heap := true;
146:C      2      saveheap := addr(saveheap, heapoff);
147:C      2      saveglob := highglob;
148:C      2      savecod := lowcod + codeoff;
149:C      2      savedef := sysdefs;
150:C      2      saveentry := entrypoint;
151:C      2      savefiles := openfileptr;      openfileptr := anyptr(-1);
152:C      2      end;
153:C      1      end;
154:S
155:D      1      procedure unload_segment;
156:C      1      begin
157:C      1      if segment_list = nil then escape(121) else
158:C      1      with segment_list^ do
159:C      1      begin
160:C      2      segment_list := savelist;
161:C      2      if restore_heap then release(saveheap);
162:C      2      highglob := saveglob;
163:C      2      lowcod := savecod;
164:C      2      sysdefs := savedef;
165:C      2      entrypoint := saveentry;
166:C      2      openfileptr := savefiles;
167:C      2      end;
168:C      1      end;
169:S
170:D      1      procedure unload_all;
171:C      1      begin
172:C      1      while segment_list <> nil do unload_segment;
173:C      1      end;
174:S
175:D      1      procedure load_seg(var filename: fid; p: proc);
176:D      -4      2      var space: integer;
177:D      -8      2      modnum: integer;
178:D      -12     2      highheap0: address;
179:D      -16     2      ESCTEMP: INTEGER;      { JWS 5/24/83 }
180:D      -20     2      IOTEMP: INTEGER;      { JWS 5/24/83 }

```

```

181:S
182:S      2      begin
183:C      2      LOADFI8.PHP:=NIL;      { JWS 5/24/83 }
184:C      2      try
185:C      3      space := memavail - 3000;      {guess as to required stack space}
186:C      3      if space <= 0 then escape(-2);
187:C      3      mark(lowheap.p);
188:C      3      highheap.a := lowheap.a + space; release(highheap.p);
189:C      3      highheap0 := highheap;
190:C      3      newmods := sysdefs;      endmod := sysdefs;
191:C      3      end;
192:S
193:C      3      openlinkfile(filename);
194:C      3      if fdirectory = nil then escape(-10);
195:C      3      for modnum := 1 to fdirectory[0].dnunfiles do
196:C      3      begin loadinfo(modnum, true, false); checkrev; end;
197:C      3      allresolved := true; matchfile;
198:C      3      if not allresolved then escape(119);
199:C      3      highheap := highheap0;
200:S
201:C      3      countcode;
202:S
203:C      3      startglobal := highglob - a5;
204:C      3      highglob := highglob - totalglobal;
205:C      3      if highglob < lowglob then escape(117);
206:C      3      zeromem(anyptr(highglob), totalglobal);
207:C      3      call(p);
208:C      3      recover begin
209:C      4      LOCKUP;      { JWS 5/24/83 }
210:C      4      ESCTEMP:=ESCAPECODE;      { JWS 5/24/83 }
211:C      4      IOTEMP:=IORESULT;      { JWS 5/24/83 }
212:C      4      CLOSEFILES;      { JWS 5/24/83 }
213:C      4      IORESULT:=IOTEMP;      { JWS 5/24/83 }
214:C      4      unload segment;
215:C      4      LOCKDOWN;      { JWS 5/24/83 }
216:C      4      escape(ESCTEMP);      { JWS 5/24/83 }
217:C      4      end;
218:C      3      end;
219:S
220:D      1      procedure release_heap;
221:C      1      begin with segment_list^ do
222:C      1      begin
223:C      2      release(saveheap);
224:C      2      restore_heap := false;
225:C      2      end;
226:C      1      end;
227:S
228:D      1      procedure local(var filename: fid; p: proc);
229:C      1      2      var state: state_rec;
230:D      -30     2      procedure loadproc;
231:C      2      begin
232:C      3      highheap.a := highheap.a - totalreloc;
233:C      3      if highheap.a < lowheap.a then escape(-2);
234:C      3      release(highheap.p);
235:C      3      startreloc := integer(allocate(totalreloc));
236:C      3      loadtext(false);
237:C      3      end;
238:C      2      end;
239:C      1      end;
240:S

```



```

241:C      3  highheap.a := highheap.a - totaldefs;
242:C      3  if highheap.a < lowheap.a then escape(-2);
243:C      3  release(highheap.p);
244:C      3  movedefs(integer(allocate(totaldefs)));
245:C      3  release_heap;
246:C      3  call(p);
247:C      3  end;
248:C
249:C      2  begin
250:C      2  save_state(state, 0, 0);
251:C      2  load_seg(filename, loadproc);
252:C      2  unload_segment;
253:C      2  end;
254:C
255:C      -122 1 procedure call_segment (filename: fid);
256:C
257:C      2  procedure callit;
258:C      -8  3  var proc_rec: trick_proc;
259:C      -12 3  modptr: moddescptr;
260:C      3  begin
261:C      3  modptr := entrypoint;
262:C      3  while modptr <> nil do with modptr^ do
263:C      5  begin
264:C      5  if startaddr <> 0 then with proc_rec do
265:C      7  begin
266:C      7  sl := 0;
267:C      7  ep := startaddr;
268:C      7  call(p);
269:C      7  end;
270:C      5  if lastmodule then modptr := nil else modptr := link;
271:C      5  end;
272:C      3  end;
273:C
274:C      2  begin
275:C      2  local(filename, callit);
276:C      2  end;
277:C
278:C      -244 1 procedure call_segment_proc(filename: fid; symbol: proc_name);
279:C      -252 2 var p: segment_proc;
280:C      -252 2
281:C      2  procedure callit; begin call(find_proc(symbol)) end;
282:C
283:C      2  begin
284:C      2  p := find_proc(symbol);
285:C      2  if exists_proc(p) then call(p)
286:C      3  else local(filename, callit);
287:C      2  end;
288:C
289:C      -122 1 procedure load_segment (filename: fid);
290:C      -126 2 var state: state_ptr;
291:C      -126 2
292:C      2  function code_space(size: integer): anyptr;
293:C      3  begin
294:C      3  if lowcod + size > highcod then escape(122)
295:C      4  else
296:C      4  begin
297:C      4  code_space := anyptr(lowcod);
298:C      4  lowcod := lowcod + size;
299:C      4  end;
300:C      3  end;

```

```

301:C
302:C      2  procedure loadproc;
303:C      3  begin
304:C      3  startreloc := integer(code_space(totalreloc));
305:C      3  loadtext(false);
306:C      3  movedefs(integer(code_space(totaldefs)));
307:C      3  release_heap;
308:C      3  end;
309:C
310:C      2  begin
311:C      2  state := code_space(sizeof(state_rec));
312:C      2  save_state(state, 0, -sizeof(state_rec));
313:C      2  load_seg(filename, loadproc);
314:C      2  end;
315:C
316:C      -122 1 procedure load_heap_segment(filename: fid);
317:C      -126 2 var state: state_ptr;
318:C
319:C      2  procedure loadproc;
320:C      3  begin
321:C      3  loadtext(true);
322:C      3  movedefs(startreloc+totalreloc);
323:C      3  release(anyptr(startdefs+totaldefs));
324:C      3  end;
325:C
326:C      2  begin
327:C      2  newbytes(state, sizeof(state_rec));
328:C      2  save_state(state, -sizeof(state_rec), 0);
329:C      2  mark(lowheap.p); startreloc := lowheap.a;
330:C      2  load_seg(filename, loadproc);
331:C      2  end;
332:C
333:C      1  end;

```

No errors. No warnings.
 ***** Nonstandard language features enabled *****

SRMAM

Description

SRMAM is the access method (transfer method) used to read and write files on the SRM.

Requirements

SRM_DRV and the I/O library drivers for the 98629 interface card (DATA_COMM).

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:      (*
2:
3:      (c) Copyright Hewlett-Packard Company, 1983.
4:      All rights are reserved. Copying or other
5:      reproduction of this program except for archival
6:      purposes is prohibited without the prior
7:      written consent of Hewlett-Packard Company.
8:
9:
10:     RESTRICTED RIGHTS LEGEND
11:
12:     Use, duplication, or disclosure by the Government
13:     is subject to restrictions as set forth in
14:     paragraph (b) (3) (B) of the Rights in Technical
15:     Data and Computer Software clause in
16:     DARF 7-104.9(a).
17:
18:     HEWLETT-PACKARD COMPANY
19:     0 Fort Collins, Colorado      *)
20:
21:
22: 0 $copyright 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'$
23: 0 $range off$
24: 0 $debug off$
25: 0 $modcall$
26:
27: 0 module srhammodule;
28:
29: 1 $search 'SRH.DRV'
30: 1 'IOLIB:KERNEL'$
31: 1 import
32: 1 sysglobals,
33: 1 misc,
34: 1 declarations,
35: 1 srh;
36:
37: 1 export
38:
39: 1 procedure srham(fp      : fibp;
40:                  request : amrequesttype;
41:                  anyvar  : buffer;
42:                  bufsize : integer;
43:                  position: integer);
44:
45: 1 implement
46: 1 type
47: 1 pointer = ^char;
48:
49: 1 (*****
50: 1 procedure srh_read(anyvar f      : fib;
51:                    startoffset : integer;
52:                    bytecount    : integer;
53:                    ramaddress    : charptr);
54:
55: 2 var
56:   sent      : integer;
57:   access    : integer;
58:   offset    : integer;
59:
60: 2 begin
61:   if bytecount > 0 then
62:     with f, packet_ptr.rhead^ (, packet_ptr.rread^ do

```

```

61: 2 begin
62:   access := random_access;
63:   offset := startoffset;
64:   sent := 0;
65:   if bytecount > 512 then
66:     begin
67:       sendreadpack(funit,fileid,access,S12,offset,ramaddress);
68:       sent := -12;
69:       ramaddress := addr(ramaddress^,S12);
70:       access := sequential_access;
71:       offset := 0;
72:
73:       while (bytecount - sent > 512) and (ioresult = ord(inoerror)) do
74:         begin
75:           sendreadpack(funit,fileid,access,S12,offset,ramaddress);
76:           sent := sent + 512;
77:           ramaddress := addr(ramaddress^,S12);
78:           packetin(funit,req_read);
79:         end;
80:
81:       if iorresult = ord(inoerror) then
82:         begin
83:           sendreadpack(funit,fileid,access,bytecount-sent,offset,ramaddress);
84:           packetin(funit,req_read);
85:         end;
86:
87:       if (bytecount > 512) and (ioresult = ord(inoerror)) then
88:         packetin(funit,req_read);
89:
90:       if iorresult <> ord(inoerror) then
91:         resetcard(funit);
92:
93:       end;
94:     end;
95:
96: 1 (*****
97: 1 procedure srh_write(anyvar f      : fib;
98:                     startoffset : integer;
99:                     bytecount    : integer;
100:                     ramaddress    : charptr);
101:
102: 2 var
103:   sent      : integer;
104:   access    : integer;
105:   offset    : integer;
106:
107: 2 begin
108:   if bytecount > 0 then
109:     with f, packet_ptr.rhead^ , packet_ptr.rwrite^ do
110:       begin
111:         access := random_access;
112:         offset := startoffset;
113:         sent := 0;
114:         if bytecount > 512 then
115:           begin
116:             sendwritepack(funit,fileid,access,S12,offset,ramaddress);
117:             sent := 512;
118:             ramaddress := addr(ramaddress^,S12);
119:             access := sequential_access;
120:             offset := 0;

```



```

121:C      8      begin
122:C      6      sendwritepack(funit,fileid,access,S12,offset,ramaddress);
123:C      6      sent      := sent + S12;
124:C      6      ramaddress := addr(ramaddress^,S12);
125:C      6      packetin(funit,req_write);
126:C      6      end;
127:C      5      end;
128:S
129:C      4      if ioresult = ord(inoerror) then
130:C      5      begin
131:C      5      sendwritepack(funit,fileid,access,bytecount-sent,offset,ramaddress);
132:C      5      packetin(funit,req_write);
133:C      5      end;
134:S
135:C      4      if (bytecount > S12) and (ioresult = ord(inoerror)) then
136:C      5      packetin(funit,req_write);
137:S
138:C      4      if ioresult <> ord(inoerror) then
139:C      5      resetcard(funit);
140:C      4      end;
141:C      2 end;
142:S
143:D      1 (*****
144:D      1 procedure srm_clearunit(anyvar f : f1b);
145:D      2 var
146:D      -2 2 u : unitnum;
147:D      -3 2 keepworkdirs : boolean;
148:C      2 begin
149:C      3 with f, unitable^[funit], isc_table[sc] do
150:C      3 if (card_id <> hp98629) then (this is not a theodore card)
151:C      4 ioresult := ord(znodevice)
152:C      4 else
153:C      4 begin
154:C      4 resetcard(funit);
155:C      4 areyoualivepack(funit);
156:C      4 if ioresult = ord(inoerror) then
157:C      5 volpack(funit);
158:C      4 if ioresult <> ord(inoerror) then
159:C      5 ioresult := ord(znodevice)
160:C      5 else
161:C      5 begin
162:C      5 keepworkdirs := false;
163:C      5 if strlen(uvid) > 0 then
164:C      6 keepworkdirs := true
165:C      6 else
166:C      6 for u := 1 to maxunit do
167:C      7 if sc = unitable^[u].sc then
168:C      8 if ba = unitable^[u].ba then
169:C      9 if strlen(unitable^[u].uvid) > 0 then
170:C      10 keepworkdirs := true;
171:C      9 gangcleanpack(funit, keepworkdirs);
172:C      5 end;
173:C      4 end;
174:C      2 end;
175:S
176:D      1 (*****
177:D      1 procedure srmam(fp : f1bp;
178:D      2 request : amrequesttype;
179:D      2 anyvar buffer : window;
180:D      2 bufsize : integer;

```

```

181:D      2 position: integer);
182:C      2 begin
183:C      2 ioresult := ord(inoerror);
184:C      2 srmsavesc := 0;
185:C      2 lockup;
186:C      2 try
187:C      3 with fp^ do
188:C      4 if (request <> clearunit) and (unitable^[funit].offline) then
189:C      5 ioresult := ord(znodevice)
190:C      5 else
191:C      5 case request of
192:C      6 readbytes :begin
193:C      6 if flockable and (not flocked) then
194:C      7 ioresult := ord(ifileunlocked)
195:C      7 else
196:C      7 srm_read(fp^,position,bufsize,addr(buffer));
197:C      6 end;
198:S
199:C      6 writebytes :begin
200:C      6 if flockable and (not flocked) then
201:C      7 ioresult := ord(ifileunlocked)
202:C      7 else
203:C      7 srm_write(fp^,position,bufsize,addr(buffer));
204:C      6 end;
205:S
206:C      6 clearunit :begin
207:C      6 srm_clearunit(fp^);
208:C      6 end;
209:S
210:C      6 flush : (do nothing, but no error);
211:S
212:C      6 otherwise ioresult := ord(ibadrequest);
213:C      6 end;
214:C      3 if ioresult = ord(isrmcatchall) then
215:C      4 if srmsavesc <> 0 then
216:C      5 escape(srmsavesc);
217:C      5 recover
218:C      3 begin
219:C      3 if escapecode = ioescapecode then
220:C      4 ioresult := ord(isrmcatchall)
221:C      4 else
222:C      4 begin
223:C      4 lockdown;
224:C      4 escape(escapecode);
225:C      4 end;
226:C      3 end;
227:C      2 lockdown;
228:C      2 end; (srmammodule)
229:S
230:C      1 end.

```

No errors. No warnings.
 ***** Nonstandard language features enabled *****

SRMDAM

Description

SRMDAM is the Directory Access Method for the SRM.

Requirements

SRM_DRV and the I/O library drivers for the 98629 interface card (DATA_COMM).

Notes

The DAM operates by making requests to the SRM to which it is connected. It does not understand the layout of an SRM disc.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1983.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S     RESTRICTED RIGHTS LEGEND
11:S
12:S     Use, duplication, or disclosure by the Government
13:S     is subject to restrictions as set forth in
14:S     paragraph (b) (3) (B) of the Rights in Technical
15:S     Data and Computer Software clause in
16:S     DAR 7-104.9(a).
17:S
18:S     HEWLETT-PACKARD COMPANY
19:D     0 Fort Collins, Colorado      *)
20:S
21:S
22:D     0 $copyright 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'S
23:D     0 $range off$
24:D     0 $debug off$
25:D     0 $modals$
26:D     0 program init_srm;
27:S
28:D     1 module srmdamodule;
29:S
30:D     1 $search 'SRM_DRV'
31:D     1 $LIB:KERNEL'S
32:D     1 import
33:D     1   sysglobals,
34:D     1   misc,
35:D     1   bootdamodule,
36:D     1   iodeclarations,
37:D     1   srm;
38:S
39:D     1 export
40:S
41:D     1 procedure srmdaminit;
42:S
43:D     1 procedure srmdam(anyvar f      : fib;
44:D     2                               unum : unitnum;
45:D     2                               request : damrequesttype);
46:S
47:D     1 implement
48:S
49:D     1 type
50:D     1   passarray = array[1..8] of passentry;
51:D     1 const
52:D     1   extentsize = 8*512; {arbitrary choice -- multiple of common block sizes}
53:D     1   constpassarray = passarray;
54:D     1
55:D     1   passentry[pb:hex('80000000'),pword:'MANAGER' ],
56:D     1   passentry[pb:hex('40000000'),pword:'READ'   ],
57:D     1   passentry[pb:hex('20000000'),pword:'WRITE'  ],
58:D     1   passentry[pb:hex('10000000'),pword:'SEARCH' ],
59:D     1   passentry[pb:hex('08000000'),pword:'PURGELINK'],
60:D     1   passentry[pb:hex('04000000'),pword:'CREATELINK'],
61:D     1   passentry[pb:hex('FFFFFFF'),pword:'ALL'   ],

```

```

61:D     1   passentry[pb:hex('00000000'),pword:'NONE' ]
62:D     1   ];
63:D     1   allcapabilities = access_capabilities[32 of true];
64:D     1   nocapabilities = access_capabilities[32 of false];
65:D     1   temp_file_pass = '>TEMP_FILE_PASS<'; {password on temporary files}
66:D     1   BDATTYPE = -5791;
67:D     1   BASICBINTYPE = -5775;
68:D     1   BASICPROGTYPE = -5808;
69:D     1   SYSTHTYPE = -5822;
70:D     1   DATATYPE = -5822;
71:D     1   COUETYPE = -5882;
72:D     1   TEXTTYPE = -5570;
73:S
74:D     1 var
75:D     -4 1 passwordarrayptr : ^passarray;
76:D     -6 1 tempcounter : shortint;
77:S
78:D     -6 1 (*****
79:D     1 procedure setioresresult(result : integer);
80:D     2 begin
81:D     2   if ioresult = ord(inoerror) then
82:D     2   ioresult := result;
83:D     2 end;
84:S
85:D     -6 1 (*****
86:D     1 function mapfkind(ftype : gang_file_codes) : filekind;
87:D     2 var
88:D     -2 2 fk : filekind;
89:D     2 begin
90:D     2   mapfkind := datafile;
91:D     2   for fk := lastkind downto untypedfile do
92:D     2   if eftable[fk] = ftype.s12 then
93:D     2   mapfkind := fk;
94:D     2 end;
95:S
96:D     -6 1 (*****
97:D     1 procedure paoc16tostr(anyvar paoc : name_type;
98:D     2 anyvar strng : string255);
99:D     2 var
100:D     -2 2 i : shortint;
101:D     2 begin
102:D     2   i := sizeof(paoc);
103:D     2   while (paoc[i] = ' ') and (i > 0) do
104:D     3   i := i - 1;
105:D     2   setstrlen(strng,0);
106:D     2   strmove(i,paoc,1,strng,1);
107:D     2 end;
108:S
109:D     -6 1 (*****
110:D     1 procedure strtopaoc16(anyvar strng : string255;
111:D     2 anyvar paoc : name_type);
112:D     2 begin
113:D     2   paoc := ' ';
114:D     2   if strlen(strng) < 17 then
115:D     2   strmove(strlen(strng),strng,1,paoc,1);
116:D     2 end;
117:S
118:D     -6 1 (*****
119:D     1 procedure setup_fns(var f : fib;
120:D     2 anyvar fns : file_name_set);

```



```

121:C      2 begin
122:C      2   with f, fns do
123:C      3   if strlen(ftid) = 0 then
124:C      4   setioresresult(ord(ibadtitle))
125:C      4   else
126:C      4   begin
127:C      4   strtopaoc16(ftid,file_name);
128:C      4   strtopaoc16(ffpw,password);
129:C      4   end;
130:C      2 end;
131:S
132:D      -6 1 (*****
133:D      1 procedure setup_fns3(var f      : fib;
134:D      2                      anyvar nsa : name_set_array_three);
135:D      2 var
136:D      2   n      : integer;
137:D      2   tempioresresult : integer;
138:D      2   tempstr : string[16];
139:C      -26 2 begin
140:C      2   with f do
141:C      3   if (not fisnew) or fnosrmtemp then
142:C      4   setup_fns(f,nsa)
143:C      4   else
144:C      4   if (strlen(ftid) = 0) and (not faronymous) then
145:C      5   setioresresult(ord(ibadtitle))
146:C      5   else
147:C      5   begin
148:C      6   with nsa[1] do
149:C      6   begin
150:C      6   password := ' ';
151:C      6   file_name := 'WORKSTATIONS';
152:C      6   end;
153:C      5   with nsa[2] do
154:C      6   begin
155:C      6   password := ' ';
156:C      6   file_name := 'TEMP_FILES';
157:C      6   end;
158:C      5   with nsa[3] do
159:C      6   begin
160:C      6   password := temp_file_pass;
161:C      6   setstrlen(tempstr,0);
162:C      6   tempioresresult := ioresult;
163:C      6   strwrite(tempstr,1,n,srmnode(unitable^[funit].sc),'_',fanonctr:1);
164:C      6   ioresult := tempioresresult;
165:C      6   strtopaoc16(tempstr,file_name);
166:C      6   end;
167:C      5   end;
168:C      2 end;
169:S
170:D      -6 1 (*****
171:D      1 procedure check_protectcode_set_array( nps : integer;
172:D      2                      var psa : protectcode_set_array);
173:D      2 {
174:S      2 9-May-1983 RAM
175:S      2 This routine has been added to check for right angle brackets ('>') in
176:S      2 passwords. If any are found, ioresult is set to ord(ibadpass). This
177:S      2 is because the parsing routines normally used with file opens terminate
178:S      2 passwords at the first '>'; therefore it is not possible to use them in
179:S      2 passwords in normal operation. If they are really desired, they can still be
180:S      2 created by calling the lower level packet routines directly.

```

```

181:S      2 Note that temporary files still have an "illegal" password.
182:S      2 This routine is called from srm_create_file and from srm_set_pass.
183:D      2 }
184:D      -4 2 var n : integer;
185:D      -8 2 i : integer;
186:C      2 begin
187:C      2   for n := 1 to nps do
188:C      3   for i := 1 to name_type_len do
189:C      4   if psa[n].password[i] = '>' then
190:C      5   setioresresult(ord(ibadpass));
191:C      2 end;
192:S
193:D      -6 1 (*****
194:D      1 procedure parseoptparm( foptstring : string255ptr;
195:D      2                      var sharemode : integer;
196:D      2                      var lockable : boolean;
197:D      2                      var nps : integer;
198:D      2                      var psa : protectcode_set_array;
199:D      2                      modeonly : boolean);
200:D      2 type
201:D      2   tokentype = (none,mode,pass,cap);
202:D      2   statetype = (needmodeorpass,needpass,needcap);
203:D      2   acstrarrtype = array [ac_manager .. ac_createlink] of string[10];
204:D      2 const
205:D      2   tokenlen = 16;
206:D      2   acstrarray = acstrarrtype['MANAGER','READ','WRITE',
207:D      2   'SEARCH','PURGELINK','CREATELINK'];
208:D      2 var
209:D      -2 2 typeoftoken : tokentype;
210:D      -4 2 state : statetype;
211:D      -8 2 sindex : integer;
212:D      -9 2 delim : char;
213:D      -12 2 ac : ac_manager .. ac_createlink;
214:D      -30 2 token : string[16];
215:D      -31 2 ok : boolean;
216:S
217:D      2 procedure getuntildelim(deli1 : char;
218:D      3                      deli2 : char);
219:D      3 var
220:D      -4 3 startindex : integer;
221:C      3 begin
222:C      3   delim := chr(0);
223:C      3   startindex := sindex;
224:C      3   while (sindex <= strlen(foptstring^)) and (delim = chr(0)) do
225:C      4   if (foptstring[sindex] <> deli1)
226:C      5   and (foptstring[sindex] <> deli2) then
227:C      5   sindex := sindex + 1;
228:C      5   else
229:C      5   delim := foptstring[sindex];
230:C      5   if (sindex - startindex) <= tokenlen then
231:C      4   token := str(foptstring^,startindex,sindex - startindex);
232:C      4   else
233:C      4   setioresresult(ord(ibadvalue));
234:C      3   if sindex <= strlen(foptstring^) then
235:C      4   sindex := sindex + 1;
236:C      3 end;
237:S
238:S      2 begin (parseoptparm)
239:C      2   sharemode := exclusive_share_code;
240:C      2   lockable := false;

```



```

241:C      2 state := needmodeorpass;
242:C      2 nps := 0;
243:C      2 sindx := 1;
244:C      2 if foptstring <> nil then
245:C      3 while (sindx <= strlen(foptstring^)) and (ioresult = ord(inoerror)) do
246:C      4 begin
247:C      5   case state of
248:C      6     needmodeorpass:
249:C      7       begin
250:C      8         getuntilldelim(':', ':');
251:C      9         if delim = ':' then
252:C      10          begin
253:C      11            typeoftoken := pass;
254:C      12            state := needcap;
255:C      13          end
256:C      14        else
257:C      15          begin
258:C      16            typeoftoken := mode;
259:C      17            state := needpass;
260:C      18          end;
261:C      19        if modeonly then
262:C      20          sindx := strlen(foptstring^) + 1;
263:C      21        end;
264:C      22      needpass : begin
265:C      23        getuntilldelim(':', ':');
266:C      24        if delim = ':' then
267:C      25          begin
268:C      26            typeoftoken := pass;
269:C      27            state := needcap;
270:C      28          end
271:C      29        else
272:C      30          setioresresult(ord(ibadvalue));
273:C      31        end;
274:C      32      needcap : begin
275:C      33        getuntilldelim(':', ':');
276:C      34        typeoftoken := cap;
277:C      35        if delim = ':' then
278:C      36          state := needcap;
279:C      37        else if delim = ':' then
280:C      38          state := needpass;
281:C      39        end;
282:C      40      end;
283:C      41    if ioresult = ord(inoerror) then
284:C      42      case typeoftoken of
285:C      43        mode : begin
286:C      44          upc(token);
287:C      45          if token = 'EXCLUSIVE' then
288:C      46            sharemode := exclusive_share_code
289:C      47          else if token = 'SHARED' then
290:C      48            sharemode := shared_share_code
291:C      49          else if token = 'LOCKABLE' then
292:C      50            begin
293:C      51              sharemode := shared_share_code;
294:C      52              lockable := true;
295:C      53            end
296:C      54          else
297:C      55            setioresresult(ord(ibadvalue));
298:C      56          end;
299:C      57        pass : begin
300:C      58          nps := nps + 1;

```

```

301:C      6 with psa[nps] do
302:C      7   begin
303:C      8     strtopaoc16(token,password);
304:C      9     capabilities := nocapabilities;
305:C      10   end;
306:C      11 end;
307:C      12 cap : begin
308:C      13   upc(token);
309:C      14   ok := false;
310:C      15   with psa[nps] do
311:C      16     if token = 'ALL' then
312:C      17       begin
313:C      18         capabilities := allcapabilities;
314:C      19         ok := true;
315:C      20       end
316:C      21     else
317:C      22       for ac := ac_manager to ac_createlink do
318:C      23         if token = acstrarray[ac] then
319:C      24           begin
320:C      25             capabilities[ac] := true;
321:C      26             ok := true;
322:C      27           end;
323:C      28         if not ok then
324:C      29           setioresresult(ord(ibadvalue));
325:C      30         end;
326:C      31       end;
327:C      32 end;
328:C      33 end;
329:C      34 end;
330:D      -6 1 (*****
331:D      1 procedure srm_close_fileid(unum : unitnum;
332:D      2 var fileid : integer);
333:C      2 begin
334:C      3   if fileid = 0 then
335:C      4     fileid := -1
336:C      5   else
337:C      6     if (fileid > 0) and (fileid <> unitable^[unum].dvrtemp) then
338:C      7       with packet_ptr.rhead^ do
339:C      8         begin
340:C      9           closepack(unum,fileid);
341:C      10          if status = 0 then
342:C      11            fileid := -1;
343:C      12          end;
344:C      13        end;
345:C      14      end;
346:D      -6 1 (*****
347:D      1 procedure srm_close_pathid(unum : unitnum;
348:D      2 var pathid : integer;
349:D      3 savepathid : boolean);
350:C      3 begin
351:C      4   if not savepathid then
352:C      5     srm_close_fileid(unum,pathid);
353:C      6   end;
354:D      -6 1 (*****
355:D      1 procedure translatedate(var srmdate : date_type;
356:D      2 var systemdate : datevec;
357:D      3 var systemtime : timerec);
358:D      3 var
359:D      4 time : integer;
360:D      4

```



```

361:C      2 begin
362:C      2   with srmdate do
363:C      3   begin
364:C      3     with systemdate do
365:C      4     begin
366:C      4       month      := date.month;
367:C      4       day        := date.day;
368:C      4       year       := date.year;
369:C      4     end;
370:C      3   with systmtime do
371:C      4   begin
372:C      4     time         := seconds_since_midnight;
373:C      4     hour         := time div 3600;
374:C      4     minute      := (time-(hour*3600)) div 60;
375:C      4     centisecond  := (time mod 60) * 100;
376:C      4   end;
377:C      3   end;
378:C      2 end;
379:S
380:D      -6 1 (*****
381:D      1 procedure srm_get_dir_info(anyvar dircatentry : catentry;
382:D      2 var dirid : integer;
383:D      2 unum : unitnum;
384:D      2 long : boolean;
385:D      2 dir_is_dvrtemp: boolean);
386:D      2 const
387:D      2   zerodate   = daterec[year:0,day:0,month:0];
388:D      2   zerotime   = timerec[hour:0,minute:0,centisecond:0];
389:D      2 var
390:D      2   n : integer;
391:D      -8   tempioresult : integer;
392:C      2 begin
393:C      2   with dircatentry, unitable^[unum] do
394:C      3   begin
395:C      3     setstrlen(cname,0);
396:C      3     volpack(unum);
397:C      3     with packet_ptr.rvol^, packet_ptr.rhead^ do
398:C      4     begin
399:C      4       if status = 0 then
400:C      5       if not exist,value then
401:C      6       setioresresult(ord(lostunit)) {set ioresult to no volume}
402:C      6     else
403:C      6       begin
404:C      6         paoc18tostr(volume_name,cname);
405:C      6         cextral := -1; {max_file_size div 32}
406:C      6         cpsize  := -1;
407:C      6         clsize  := -1;
408:C      6         cextra2 := interleave;
409:C      6         cstart  := -1;
410:C      6         cblocksize := 1;
411:C      6         ccreatedate := zerodate;
412:C      6         ccreatetime := zerotime;
413:C      6         clastdate  := zerodate;
414:C      6         clasttime  := zerotime;
415:C      6         setstrlen(cinfo,0);
416:C      6         tempioresresult := ioresult;
417:C      6         strwrite(cinfo,1,n,'SRM ',5c:1,',','ba:1,',',du:1);
418:C      6         ioresult := tempioresresult;
419:C      6         if dirid > 0 then
420:C      7         begin

```

```

421:C      7   fileinfo^pack(unum,dirid);
422:C      7   with packet_ptr.rfileinfo^, file_info do
423:C      8   if status <> 0 then
424:C      9   begin
425:C      9     ioresult := tempioresresult;
426:C      9     if dir_is_dvrtemp then
427:C      10    dirid := 0;
428:C      9   end
429:C      9   else
430:C      8   begin
431:C      8     if file_name <> ' ' then
432:C      10    paoc18tostr(file_name,cname);
433:C      9     if long then
434:C      10    begin
435:C      10      translatedate(creation_date,ccreatedate,ccreatetime);
436:C      10      translatedate(last_access_date,clastdate,clasttime);
437:C      10    end;
438:C      7   end; end;
439:C      7   end;
440:C      6   end; end;
441:C      4   end;
442:C      3   end;
443:C      2 end;
444:S
445:D      -6 1 (*****
446:D      1 procedure srm_get_vol_name(anyvar f : vid;
447:D      2 unum : unitnum);
448:D      2 var
449:D      -80   dircatentry : catentry;
450:C      2 begin
451:C      2   srm_get_dir_info(dircatentry,unitable^[unum].dvrtemp,unum,false,true);
452:C      2   f := dircatentry.cname;
453:C      2 end;
454:S
455:D      -6 1 (*****
456:D      1 procedure srm_set_pass(anyvar f : fib;
457:D      2 unum : unitnum);
458:D      2 type
459:D      2   catarray = array[0..maxint] of passentry;
460:D      2   catarrayptr = ^catarray;
461:D      2 var
462:D      -16   volpass : name_type;
463:D      -52   fns : file_name_set;
464:D      -56   nps : integer;
465:D      -57   done : boolean;
466:D      -62   i : integer;
467:D      -66   j : integer;
468:D      -70   catindx : integer;
469:D      -74   catentryindx : integer;
470:D      -74   tempcapbits : record case boolean of
471:D      -74   true : (i : integer);
472:D      -74   false : (b : access_capabilities);
473:D      -78   end;
474:D      -654   psa : protectcode_set_array;
475:C      2 begin
476:C      2   with f do
477:C      3   begin
478:C      3     setup_fns(f,fns);
479:C      3     if ioresult = ord(inoerror) then
480:C      4     begin

```



```
481:C      4      for i := 1 to fpeof do
482:C      5          with psa[i], catarrayptr(fwindow)^[i-1] do
483:C      6              begin
484:C      6                  strtopaoc16(pword,password);
485:C      6                  tempcapbits.i := pbits;
486:C      6                  capabilities := tempcapbits.b;
487:C      6                  nps := 1;
488:C      6              end;
489:C      6          strtopaoc16(fvid,volpass);
490:C      4          check_protectcode_set_array(nps,psa);
491:C      4          if ioreult = ord(inoerror) then
492:C      5              changeprotectpack(unum,1,addr(fns),start_alternate,pathid,
493:C      5                  volpass,nps,addr(psa));
494:C      4          end;
495:C      3      end;
496:C      2  end;
497:C      1
498:D      -6  1  (*****
499:D      1  procedure srm_cat_pass(anyvar f      : fib;
500:D      2      unum      : unitnum);
501:D      2  type
502:D      2      catarray      = array[0..maxint] of passentry;
503:D      2      catarrayptr  = ^catarray;
504:D      2  var
505:D      -16  volpass      : name_type;
506:D      -52  fns      : file_name_set;
507:D      -53  done      : boolean;
508:D      -58  i      : integer;
509:D      -62  j      : integer;
510:D      -66  catindx      : integer;
511:D      -70  catentryindx : integer;
512:D      -70  tempcapbits : record case boolean of
513:D      -70      true : (i : integer);
514:D      -70      false : (b : access_capabilities);
515:D      -74  end;
516:C      2  begin
517:C      2      catentryindx := 0;
518:C      2      done := false;
519:C      2      with f, packet_ptr.rcatpass^ do
520:C      3          begin
521:C      3              setup_fns(f, fns);
522:C      3              if ioreult = ord(inoerror) then
523:C      4                  begin
524:C      4                      strtopaoc16(fvid,volpass);
525:C      4                      foptstring := anyptr(passwordarrayptr);
526:C      4                      catindx := fpos + 1;
527:C      4                      while (catentryindx < fpeof) and (not done) and (ioreult = ord(inoerror)) do
528:C      5                          begin
529:C      5                              catpasspack(unum,1,addr(fns),start_alternate,
530:C      5                                  pathid,volpass,24,catindx);
531:C      5                              if ioreult = ord(inoerror) then
532:C      6                                  begin
533:C      6                                      i := 1;
534:C      6                                      if actual_num_passwords < 24 then
535:C      7                                          done := true;
536:C      6                                      while i <= actual_num_passwords do
537:C      7                                          if catentryindx < fpeof then
538:C      8                                              begin
539:C      8                                                  with password_info[i], catarrayptr(fwindow)^[catentryindx] do
540:C      9                                                      begin
```

```
541:C      8          paoc16tostr(password,pword);
542:C      8          tempcapbits.b := capabilities;
543:C      8          pbits := tempcapbits.i;
544:C      8          end;
545:C      8          i := i + 1;
546:C      8          catentryindx := catentryindx + 1;
547:C      8          end
548:C      8      else
549:C      8          begin
550:C      8              i := 25;
551:C      8              done := true;
552:C      8          end;
553:C      6          catindx := catindx + 24;
554:C      6          end;
555:C      5          end;
556:C      4          end;
557:C      3          fpeof := catentryindx;
558:C      3          end;
559:C      2  end;
560:C      1
561:D      -8  1  (*****
562:D      1  procedure srm_catalog(anyvar f      : fib;
563:D      2      unum      : unitnum);
564:D      2  type
565:D      2      catarray      = array[0..maxint] of catentry;
566:D      2      catarrayptr  = ^catarray;
567:D      2      ac_char_arr  = array [ac_manager..ac_createlink] of char;
568:D      2  const
569:D      2      ac_chars      = ac_char_arr['M','R','W','S','P','C'];
570:D      2  var
571:D      -16  volpass      : name_type;
572:D      -52  fns      : file_name_set;
573:D      -53  done      : boolean;
574:D      -58  i      : integer;
575:D      -62  j      : integer;
576:D      -66  catindx      : integer;
577:D      -70  catentryindx : integer;
578:D      -72  ac      : access_code_type;
579:C      2  begin
580:C      2      catentryindx := 0;
581:C      2      done := false;
582:C      2      with f, packet_ptr.rcat^ do
583:C      3          begin
584:C      3              strtopaoc16(fvid,volpass);
585:C      3              catindx := fpos + 1;
586:C      3              while (catentryindx < fpeof) and not done do
587:C      4                  begin
588:C      4                      catpack(unum,0,addr(fns),start_alternate,
589:C      4                          pathid,volpass,7,catindx);
590:C      4                      if ioreult <> ord(inoerror) then
591:C      5                          done := true;
592:C      4                      else
593:C      5                          begin
594:C      5                              i := 1;
595:C      5                              if actual_num_files < 7 then
596:C      6                                  done := true;
597:C      5                              while i <= actual_num_files do
598:C      6                                  if catentryindx < fpeof then
599:C      7                                      begin
600:C      7                                          with cat_info[i], catarrayptr(fwindow)^[catentryindx] do
```



```

601:C      8      begin
602:C      8      paoc16tostr(file_name,cname);
603:C      8      cefit      := file_code.s12;
604:C      8      ckind      := mapfkind(file_code);
605:C      8      cpsize      := physical_size;
606:C      8      clsize      := logical_eof;
607:C      8      cstart      := -1;
608:C      8      translatedate(creation_date,ccreatedate,ccreateetime);
609:C      8      translatedate(last_access_date,clastdate,clasttime);
610:C      8      cblocksiz:= -1;
611:C      8      cextra1      := -1;
612:C      8      cextra2      := -1;
613:C      8      setstrlen(cinfo,ord(ac_createlink)+1);
614:C      8      for ac := ac_manager to ac_createlink do
615:C      9      if capabilities[ac] then
616:C      10      cinfo[ord(ac) + 1] := ac_chars[ac]
617:C      10      else
618:C      10      cinfo[ord(ac) + 1] := ' ';
619:C      9      case share_code of
620:C      9      exclusive_share_code : cinfo := cinfo + ' EXCLUSIVE';
621:C      9      shared_share_code   : cinfo := cinfo + ' SHARED';
622:C      9      {
623:C      9      closed_share_code    : cinfo := cinfo + ' CLOSED';
624:C      9      }
625:C      9      corrupt_share_code   : cinfo := cinfo + ' CORRUPT';
626:C      9      otherwise          : cinfo := cinfo + ' CLOSED';
627:C      9      end;
628:C      8      i := i + 1;
629:C      7      catentryindx := catentryindx + 1;
630:C      7      end
631:C      7      else
632:C      7      begin
633:C      7      := 8;
634:C      7      done := true;
635:C      7      end;
636:C      7      catindx := catindx + 7;
637:C      5      end;
638:C      5      fpeof := catentryindx;
639:C      4      end;
640:C      3      end;
641:C      2      end;
642:C      2      end;
643:S      -6      1 (*****
644:D      1      procedure srm_open_dir(anyvar f : fib;
645:D      2      unum : unitnum;
646:D      2      opentype : gang_open_type;
647:D      2      openparent:boolean);
648:D      2      var
649:D      2      volpass : name_type;
650:D      -16     2      lentitle : integer;
651:D      -20     2      sindx : integer;
652:D      -24     2      pindx : integer;
653:D      -28     2      nindx : integer;
654:D      -32     2      l : integer;
655:D      -36     2      path : path_start_type;
656:D      -40     2      saveid : file_id_type;
657:D      -42     2      origpathid : file_id_type;
658:D      -46     2      last : boolean;
659:D      -47     2      alreadyopen : boolean;
660:D      -48     2

```

```

661:D      -52     2      nfns : integer;
662:D      -268    2      nsa : name_set_array;
663:S      2
664:D      2      procedure getpaoc(anyvar paoc : name_type;
665:D      2      dell : char;
666:D      3      del2 : char);
667:D      3      var
668:D      -1      3      done : boolean;
669:C      3      begin
670:C      3      with f do
671:C      4      begin
672:C      4      done := false;
673:C      4      while (sindx < lentitle) and (pindx <= name_type.len) and (not done) do
674:C      5      if (ftitle[sindx] = dell) or (ftitle[sindx] = del2) then
675:C      6      done := true;
676:C      6      else
677:C      6      begin
678:C      6      paoc[pindx] := ftitle[sindx];
679:C      6      pindx := pindx + 1;
680:C      6      sindx := sindx + 1;
681:C      6      end;
682:C      4      if (sindx > lentitle) then
683:C      5      begin
684:C      5      if dell = '>' then
685:C      6      ioresult := ord(ibadpass);
686:C      5      end
687:C      5      else
688:C      5      begin
689:C      5      if (ftitle[sindx] <> dell) and (ftitle[sindx] <> del2) then
690:C      6      begin
691:C      6      if dell = '>' then
692:C      7      ioresult := ord(ibadpass);
693:C      7      else
694:C      7      ioresult := ord(ibadtitle);
695:C      6      end
696:C      6      else
697:C      6      if (dell = '>') then
698:C      7      if (ftitle[sindx] <> '>') then
699:C      8      ioresult := ord(ibadpass);
700:C      5      end;
701:C      4      end;
702:C      3      end;
703:S      2
704:S      2
705:S      2
706:C      2      begin
707:C      2      last := false;
708:C      2      alreadyopen := false;
709:C      2      sindx := 1;
710:S      2
711:C      2      with f do
712:C      3      begin
713:C      3      origpathid := pathid;
714:C      3      lentitle := strlen(ftitle);
715:C      3      setstrlen(ftid,0);
716:C      3      if pathid = -1 then
717:C      4      begin
718:C      4      setstrlen(fvid,0);
719:C      4      setstrlen(ffpw,0);
720:C      4      end;

```



```
721:C      3      if (sindx <= lentitle) then
722:C      4      if ftitle[sindx] = '<' then      (get volume password)
723:C      5      begin
724:C      6      sindx := sindx + 1;
725:C      5      pindx := 1;
726:C      5      volpass := '';
727:C      5      getpaoc(volpass,'>','>');
728:C      5      paoc16tostr(volpass,fvid);
729:C      5      sindx := sindx + 1;
730:C      5      end;
731:S
732:C      3      path := start_alterate;
733:C      3      if ioresult = ord(inoerror) then
734:C      4      if (sindx <= lentitle) then
735:C      5      if ftitle[sindx] = '/' then
736:C      6      begin
737:C      7      path := start_root;
738:C      6      sindx := sindx + 1;
739:C      6      end;
740:C      3      if pathid = -1 then
741:C      4      if path = start_root then
742:C      5      pathid := 0
743:C      5      else
744:C      5      pathid := unitable^[unum].dvrtemp;
745:S
746:C      3      if sindx > lentitle then
747:C      4      begin
748:C      4      last := true;
749:C      4      setstrlen(ftitle,0);
750:C      4      end
751:C      4      else
752:C      4      if (ftitle[lentitle] = '/') then
753:C      5      setioresresult(ord(ibadtitle));
754:S
755:C      3      with packet_ptr.ropen^ do
756:C      4      while (not last) and (ioresult = ord(inoerror)) do
757:C      5      begin
758:C      6      nfns := 0;
759:C      5      while (sindx <= lentitle) and (nfns < 6) and (ioresult = ord(inoerror)) do
760:C      6      begin
761:C      7      with nsa[nfns+1] do
762:C      7      begin
763:C      7      if (ftitle[sindx] = '/') then
764:C      8      ioresult := ord(ibadtitle)
765:C      8      else
766:C      8      begin
767:C      9      file_name := '';
768:C      8      password := '';
769:C      8      pindx := 1;
770:C      8      getpaoc(file_name,'/','/');
771:C      8      nindx := pindx;
772:C      8      if ioresult = ord(inoerror) then
773:C      9      if (sindx <= lentitle) then
774:C      10      if (ftitle[sindx] = '<') then
775:C      11      begin
776:C      11      sindx := sindx + 1;
777:C      11      pindx := 1;
778:C      11      getpaoc(password,'>','>');
779:C      11      sindx := sindx + 1;
780:C      11      if sindx <= lentitle then
```

```
781:C      12      if ftitle[sindx] <> '/' then
782:C      13      if nindx > lentitle then
783:C      14      ioresult := ord(ibadtitle)
784:C      14      else
785:C      14      begin
786:C      14      pindx := nindx;
787:C      14      getpaoc(file_name,'/','/');
788:C      14      end;
789:C      11      end;
790:C      11      if ioresult = ord(inoerror) then
791:C      9      if (sindx > lentitle) then
792:C      10      begin
793:C      10      last := true;
794:C      10      setstrlen(ftitle,0);
795:C      10      end;
796:C      8      end;
797:C      7      end;
798:C      6      if ioresult = ord(inoerror) then
799:C      7      nfns := nfns + 1;
800:C      6      sindx := sindx + 1;
801:C      6      end;
802:S
803:C      5      if ioresult = ord(inoerror) then
804:C      6      begin
805:C      7      if not (last and openparent) then
806:C      7      begin
807:C      7      openpack(unum,nfns,addr(nsa),path,
808:C      7      pathid,volpass,shared_share_code,opentype);
809:C      7      if ioresult <> ord(inoerror) then
810:C      8      begin
811:C      8      if last then
812:C      9      begin
813:C      9      ioresult := ord(inoerror);
814:C      9      openparent := true;
815:C      9      end;
816:C      8      end
817:C      8      else
818:C      8      begin
819:C      9      saveid := file_id;
820:C      8      if alreadyopen then
821:C      9      srm_close_pathid(unum,pathid,false)
822:C      9      else
823:C      9      alreadyopen := true;
824:C      8      pathid := saveid;
825:C      8      path := start_alterate;
826:C      8      end;
827:C      7      end;
828:C      6      if ioresult = ord(inoerror) then
829:C      7      begin
830:C      7      if last and openparent then
831:C      8      begin
832:C      8      if (nfns <= 1) then
833:C      9      begin
834:C      9      if pathid = -1 then
835:C      10      setioresresult(ord(inodirectory));
836:C      9      end
837:C      8      else
838:C      9      begin
839:C      9      openpack(unum,nfns-1,addr(nsa),path,
840:C      9      pathid,volpass,shared_share_code,opentype);
```



```

841:C      9      if ioresult <> ord(inoerror) then
842:C      10      begin
843:C      10      if alreadyopen then
844:C      11      srm_close_pathid(unum,pathid,false);
845:C      10      end
846:C      10      else
847:C      10      begin
848:C      10      saveid := file_id;
849:C      10      if alreadyopen then
850:C      11      srm_close_pathid(unum,pathid,false);
851:C      10      pathid := saveid;
852:C      10      end;
853:C      9      end;
854:C      8      if ioresult = ord(inoerror) then
855:C      9      with nsa[nfns] do
856:C      10      begin
857:C      10      paoc16tostr(file_name,ftid);
858:C      10      ftitle := ftitle;
859:C      10      if password <> ' ' then
860:C      11      paoc16tostr(password,ffpw);
861:C      10      end;
862:C      8      end;
863:C      7      end;
864:C      6      end;
865:C      5      end;
866:C      3      if ((origpathid <> -1) and (origpathid = pathid))
867:C      4      or (pathid = unitable^[unum].dvrtemp) then
868:C      4      fsavepathid := true
869:C      4      else
870:C      4      fsavepathid := false;
871:C      3      end;
872:C      2      end;
873:S
874:D      -6 1 (*****
875:D      1 procedure srm_set_unit_prefix(anyvar f : fib;
876:D      2      unum : unitnum);
877:D      2 var
878:D      2   savpathid : integer;
879:C      2   begin
880:C      2   with f, unitable^[unum] do
881:C      3   begin
882:C      3   srm_open_dir(f,unum,open_protected_directory,false);
883:C      3   if ioresult = ord(inoerror) then
884:C      4   begin
885:C      4   if strlen(ftitle) > 0 then
886:C      5   setioresult(ord(inounit))
887:C      5   else
888:C      5   begin
889:C      5   savpathid := pathid;
890:C      5   pathid := dvrtemp;
891:C      5   dvrtemp := savpathid;
892:C      5   end;
893:C      4   srm_close_pathid(unum,pathid,false);
894:C      4   end;
895:C      3   srm_get_vol_name(uvid,unum);
896:C      3   end;
897:C      2   end;
898:S
899:D      -6 1 (*****
900:D      1 procedure doopenpack(unum : unitnum);

```

```

901:D      2      var f : fib;
902:D      2      nfns : integer;
903:D      2      anyvar nsa : name_set_array;
904:D      2      path : path_start_type;
905:D      2      volpass : name_type;
906:D      2      sharecode : integer;
907:D      -16     lockable : boolean;
908:D      -16     type
909:D      -16     trickrec = record case boolean of
910:D      -16     true : (i : integer);
911:D      -16     false : (chs : packed array [1..2] of char;
912:D      -16     si2 : shortint);
913:D      -16     end;
914:D      -16     var
915:D      -20     temprec : trickrec;
916:C      2   begin
917:C      2   with f, packet_ptr.ropen^ do
918:C      3   begin
919:C      3   if lockable and fistextvar then
920:C      4   setioresult(ord(inotlockable))
921:C      4   else
922:C      4   openpack(unum,nfns,addr(nsa),path,pathid,volpass,sharecode,open_data);
923:C      3   if ioresult = ord(inoerror) then
924:C      4   if file_code.si2 = 3 then (directory)
925:C      5   begin
926:C      5   setioresult(ord(inotondir));
927:C      5   closepack(unum,file_id);
928:C      5   end
929:C      5   else
930:C      5   begin
931:C      5   fileid := file_id;
932:C      5   fpeof := open_logical_eof;
933:C      5   fleof := open_logical_eof;
934:C      5   fleft := file_code.si2;
935:C      5   fkind := mapfkind(file_code);
936:C      5   flockable := lockable;
937:C      5   flocked := not lockable; {default to locked unless lockable}
938:C      5   if (fleft = BDATTYPE) then {BDAT file}
939:C      6   or (fleft = BASICBINTYPE) then {BIN file}
940:C      6   or (fleft = BASICPROGTYPE) then {PROG file}
941:C      6   begin
942:C      6   temprec.chs := ' ';
943:C      6   temprec.si2 := max_record_size div 2;
944:C      6   fstartaddress := temprec.i;
945:C      6   end
946:C      6   else
947:C      6   fstartaddress := boot_start_address;
948:C      6   if not fbuffered then am := amtable^[untypedfile]
949:C      6   else if fistextvar then am := amtable^[fkind]
950:C      7   else am := amtable^[datafile];
951:C      5   end;
952:C      3   end;
953:C      2   end;
954:S
955:D      -6 1 (*****
956:D      1 procedure srm_open_file(anyvar f : fib;
957:D      2      unum : unitnum);
958:D      2 type
959:D      2   trickrec = record case boolean of
960:D      2   true : (i : integer);

```



```
961:D      2      false : (si1 : shortint;
962:D      2      si2 : shortint);
963:D      2      end;
964:D      2      var
965:D      2      volpass      : name_type;
966:D      2      fns          : file_name_set;
967:D      2      temprec      : trickrec;
968:D      2      sharemode     : integer;
969:D      2      nps          : integer;
970:D      2      psa         : protectcode_set_array;
971:D      2      lockable     : boolean;
972:D      2      begin
973:D      2      with f do
974:D      2      begin
975:D      2      setup_fns(f, fns);
976:D      2      parseoptparm(foptstring,sharemode,lockable,nps,psa,true);
977:D      2      if ioresult = ord(inoerror) then
978:D      2      begin
979:D      2      strtopaoc16(fvid,volpass);
980:D      2      doopenpack(unum,f,1,fns,start_alternate,volpass,sharemode,lockable);
981:D      2      end;
982:D      2      end;
983:D      2      end;
984:D      2
985:D      -6      1 (*****
986:D      1      procedure srm_create_dir (anyvar f      : fib;
987:D      2      unum      : unitnum);
988:D      2
989:D      2      type
990:D      2      catentryptr = ^catentry;
991:D      2      const
992:D      2      dirfilecode = gang_file_codes[i:3];
993:D      2      var
994:D      2      volpass      : name_type;
995:D      2      fns          : file_name_set;
996:D      2      begin
997:D      2      with f, catentryptr(fwindow)^ do
998:D      2      if strlen(cname) = 0 then
999:D      2      setioresult(ord(1badtitle))
1000:D      2      else
1001:D      2      begin
1002:D      2      with fns do
1003:D      2      begin
1004:D      2      password      := ' ';
1005:D      2      strtopaoc16(cname,file_name);
1006:D      2      end;
1007:D      2      strtopaoc16(fvid,volpass);
1008:D      2      createpack(unum,i,addr(fns),start_alternate,pathid,volpass,0,nil,
1009:D      2      dirfilecode,directory_records,0,0,0,0);
1010:D      2      end;
1011:D      2      end;
1012:D      2
1013:D      -6      1 (*****
1014:D      1      procedure srm_create_file(anyvar f      : fib;
1015:D      2      unum      : unitnum);
1016:D      2
1017:D      2      const
1018:D      2      dirfilecode = gang_file_codes[i:3];
1019:D      2      type
1020:D      2      trickrec      = record case boolean of
1021:D      2      true : (i : integer);
1022:D      2      false : (si1 : shortint;
```

```
1021:D      2      si2 : shortint);
1022:D      2      end;
1023:D      2      var
1024:D      2      volpass      : name_type;
1025:D      2      nsa          : name_set_array_three;
1026:D      2      ext1         : integer;
1027:D      2      temprec      : trickrec;
1028:D      2      sharemode     : integer;
1029:D      2      maxrec       : integer;
1030:D      2      nps          : integer;
1031:D      2      useleft       : gang_file_codes;
1032:D      2      psa         : protectcode_set_array;
1033:D      2      i           : integer;
1034:D      2      ac          : ac_manager..ac_purgelink;
1035:D      2      lockable     : boolean;
1036:D      2
1037:D      2      begin
1038:D      2      with f, nsa[3] do
1039:D      2      begin
1040:D      2      strtopaoc16(fvid,volpass);
1041:D      2      repeat
1042:D      2      ioresult      := ord(inoerror);
1043:D      2      fanonctr      := tempcounter;
1044:D      2      tempcounter    := tempcounter + 1;
1045:D      2      useleft.1      := left;
1046:D      2
1047:D      2      if left = BDATTYPE then      (BDAT file)
1048:D      2      begin
1049:D      2      temprec.i      := fstartaddress;
1050:D      2      maxrec         := temprec.si2 * 2;
1051:D      2      if maxrec < 1 then
1052:D      2      maxrec := 1;
1053:D      2      end
1054:D      2      else
1055:D      2      maxrec         := 256;
1056:D      2      if fpos > 0 then
1057:D      2      ext1 := fpos
1058:D      2      else
1059:D      2      ext1 := extentsize;
1060:D      2      parseoptparm(foptstring,sharemode,lockable,nps,psa,false);
1061:D      2      check_protectcode_set_array(nps,psa);
1062:D      2      if (nps > 0) and (nps < 24) and (ioresult = ord(inoerror)) then
1063:D      2      begin
1064:D      2      nps := nps + 1;
1065:D      2      with psa[nps] do
1066:D      2      begin
1067:D      2      password      := temp_file_pass;
1068:D      2      capabilities   := nocapabilities;
1069:D      2      if nps > 1 then
1070:D      2      for i := 1 to nps-1 do
1071:D      2      for ac := ac_manager to ac_purgelink do
1072:D      2      if psa[i].capabilities[ac] then
1073:D      2      capabilities[ac] := true;
1074:D      2      end;
1075:D      2      end;
1076:D      2
1077:D      2      if (not fanonymous) and (ioresult = ord(inoerror)) then
1078:D      2      begin
1079:D      2      setup_fns(f,nsa);
1080:D      2      foldfileid := -1;
```



```

1081:C      5      openpack(unum,1,addr(nsa),start_alternate,pathid,volpass,sharemode,open_data);
1082:C      5      if ioresult = ord(inoerror) then
1083:C      6      begin
1084:C      6      ioresult := ord(inoerror);
1085:C      6      nsa[1].password := ' ';
1086:C      6      createpack(unum,1,addr(nsa),start_alternate,pathid,volpass,
1087:C      6      nps,addr(psa),useleft,data_records,
1088:C      6      maxrec,ext1,extentsize,fstartaddress);
1089:C      6      if ioresult = ord(inoerror) then
1090:C      6      if feft <> SYSTHYPE then
1091:C      8      fnosrmtemp := true
1092:C      8      else
1093:C      8      begin
1094:C      8      (SYSTM files must go through temp first)
1095:C      8      nsa[1].password := temp_file_pass;
1096:C      8      openpack(unum,1,addr(nsa),start_alternate,pathid,
1097:C      8      volpass,sharemode,open_data);
1098:C      6      end;
1099:C      5      if (ioresult = ord(inoerror)) and (not fnosrmtemp) then
1100:C      6      begin
1101:C      6      foldfileid := packet_ptr.ropen^.file_id;
1102:C      6      fileinfofpack(unum,foldfileid);
1103:C      6      with packet_ptr.rfileinfo^.file_info do
1104:C      7      begin
1105:C      7      if (not capabilities(ac_manager))
1106:C      8      and (not capabilities(ac_purjelink)) then
1107:C      8      setioresult(ord(ibadpass)); (won't be able to purge old)
1108:C      8      else
1109:C      8      (this test added in version 2.2 on 4-May-83)
1110:C      9      if file_code.si2 = 3 then (disallow rewrite on directory)
1111:C      7      setioresult(ord(inotondir));
1112:C      7      if ioresult <> ord(inoerror) then
1113:C      8      srm_close_fileid(unum,foldfileid);
1114:C      6      end;
1115:C      5      end;
1116:C      5      if (ioresult = ord(inoerror)) and (not fnosrmtemp) then
1117:C      4      begin
1118:C      4      setup_fns3(f,nsa);
1119:C      4      if ioresult = ord(inoerror) then
1120:C      6      begin
1121:C      6      password := ' ';
1122:C      6      createpack(unum,3,addr(nsa),start_root,pathid,volpass,
1123:C      6      nps,addr(psa),useleft,data_records,
1124:C      6      maxrec,ext1,extentsize,fstartaddress);
1125:C      6      if ioresult = ord(inoerror) then
1126:C      7      begin
1127:C      7      ioresult := ord(inoerror);
1128:C      7      createpack(unum,2,addr(nsa),start_root,pathid,volpass,
1129:C      7      0,nil,dirfilecode,directory_records,0,0,0,0);
1130:C      7      if ioresult <> ord(inoerror) then
1131:C      7      ioresult := ord(inneedtmpdir);
1132:C      8      else
1133:C      8      createpack(unum,3,addr(nsa),start_root,pathid,volpass,
1134:C      8      nps,addr(psa),useleft,data_records,
1135:C      8      maxrec,ext1,extentsize,fstartaddress);
1136:C      7      end
1137:C      7      else
1138:C      7      if ioresult = ord(idupfile) then
1139:C      7      begin
1140:C      8

```

```

1141:C      8      ioresult := ord(inoerror);
1142:C      8      password := temp_file_pass;
1143:C      8      purgepack(unum,3,addr(nsa),start_root,pathid,volpass);
1144:C      8      ioresult := ord(inoerror);
1145:C      8      password := ' ';
1146:C      8      createpack(unum,3,addr(nsa),start_root,pathid,volpass,
1147:C      8      nps,addr(psa),useleft,data_records,
1148:C      8      maxrec,ext1,extentsize,fstartaddress);
1149:C      6      end;
1150:C      5      end;
1151:C      4      until ioresult <> ord(idupfile));
1152:C      3      if ioresult <> ord(inoerror) then
1153:C      4      srm_close_fileid(unum,foldfileid)
1154:C      4      else
1155:C      4      if fnosrmtemp then
1156:C      5      begin
1157:C      5      nsa[1].password := temp_file_pass;
1158:C      5      doopenpack(unum,f,1,nsa,start_alternate,volpass,sharemode,lockable);
1159:C      5      end
1160:C      5      else
1161:C      5      begin
1162:C      5      password := temp_file_pass;
1163:C      5      doopenpack(unum,f,3,nsa,start_root,volpass,exclusive_share_code,lockable);
1164:C      5      end;
1165:C      3      end;
1166:C      2      end;
1167:C      1
1168:C      1
1169:D      -6 1 (*****
1170:D      1 procedure srm_change_name(anyvar f      : fib;
1171:D      2                               unum      : unitnum);
1172:D      2 type
1173:D      2     fidptr      = ^fid;
1174:D      2 var
1175:D      2     -16 volpass      : name_type;
1176:D      2     -18 path1         : path_start_type;
1177:D      2     -22 nfns1         : integer;
1178:D      2     -58 fns1          : file_name_set;
1179:D      2     -80 path2         : path_start_type;
1180:D      2     -64 nfns2         : integer;
1181:D      2     -100 fns2         : file_name_set;
1182:C      2 begin
1183:C      2 with f, unitable^[unum] do
1184:C      3 begin
1185:C      3 srm_open_dir(f,unum,open_directory,true);
1186:C      3 if ioresult = ord(inoerror) then
1187:C      3 setup_fns(f, fns1);
1188:C      3 if ioresult = ord(inoerror) then
1189:C      4 with fns2 do
1190:C      5 begin
1191:C      5 file_name := ' ';
1192:C      5 password := ' ';
1193:C      5 ***WARNING: (line 1193): STRPOS does not conform to HP standard, see $SWITCH_STRPOS$
1194:C      5 if strpos('/',fidptr(fwindow)^) = 0 then
1195:C      6 ***WARNING: (line 1194): STRPOS does not conform to HP standard, see $SWITCH_STRPOS$
1196:C      6 if (strpos('/',fidptr(fwindow)^) = 0) then
1197:C      7 strtopaci16(fidptr(fwindow)^,file_name);
1198:C      6 if file_name = ' ' then
1199:C      6 setioresult(ord(ibadtitle));
1200:C      6 end;

```



```
1199:C      3      if ioreresult = ord(inoerror) then
1200:C      4      begin
1201:C      4          strttopacc16(fvid,volpass);
1202:C      4          createlinkpack(unum,1,addr(fns1),start_alternate,pathid,volpass,
1203:C      4          1,addr(fns2),start_alternate,pathid,volpass,true);
1204:C      4      end;
1205:C      3      srm_close_pathid(unum,pathid,fsavepathid);
1206:C      2      end;
1207:C      2      end;
1208:S
1209:D      -6 1 (*****
1210:D      1 procedure srm_dup_link(anyvar f      : fib;
1211:D      2                                unum      : unitnum);
1212:D      2 var
1213:D      2     volpass      : name_type;
1214:D      2     volpass2     : name_type;
1215:D      2     path1        : path_start_type;
1216:D      2     nfns1        : integer;
1217:D      2     fns1         : file_name_set;
1218:D      2     path2        : path_start_type;
1219:D      2     nfns2        : integer;
1220:D      2     fns2         : file_name_set;
1221:D      -116 2 begin
1222:C      2     with f, unitable^[unum] do
1223:C      3     begin
1224:C      3         srm_open_dir(f,unum,open_directory,true);
1225:C      3         if ioreresult = ord(inoerror) then
1226:C      4         begin
1227:C      4             setup_fns(f, fns1);
1228:C      4             if ioreresult = ord(inoerror) then
1229:C      3             srm_open_dir(fibp(fwindow)^unum,open_directory,true);
1230:C      4             if ioreresult = ord(inoerror) then
1231:C      3             setup_fns(fibp(fwindow)^fns2);
1232:C      4             if ioreresult = ord(inoerror) then
1233:C      4             begin
1234:C      4                 strttopacc16(fvid,volpass);
1235:C      4                 strttopacc16(fibp(fwindow)^fvid,volpass2);
1236:C      4                 createlinkpack(unum,1,addr(fns1),start_alternate,pathid,volpass,
1237:C      4                 1,addr(fns2),start_alternate,fibp(fwindow)^pathid,volpass2,
1238:C      4                 fpurgeoldlink);
1239:C      3             srm_close_pathid(unum,pathid,fsavepathid);
1240:C      3             srm_close_pathid(unum,fibp(fwindow)^pathid,fibp(fwindow)^fsavepathid);
1241:C      2             end;
1242:C      2         end;
1243:S
1244:D      -6 1 (*****
1245:D      1 procedure srm_purge_name(anyvar f      : fib;
1246:D      2                                unum      : unitnum);
1247:D      2 var
1248:D      2     volpass      : name_type;
1249:D      -52 2     fns         : file_name_set;
1250:C      2     begin
1251:C      2         with f do
1252:C      3         begin
1253:C      3             setup_fns(f, fns);
1254:C      3             if ioreresult = ord(inoerror) then
1255:C      4             begin
1256:C      4                 strttopacc16(fvid,volpass);
1257:C      4                 purgepack(unum,1,addr(fns),start_alternate,pathid,volpass);
1258:C      4             end;
```

```
1259:C      3      end;
1260:C      2      end;
1261:S
1262:D      -6 1 (*****
1263:D      1 procedure srm_purge_file(anyvar f      : fib;
1264:D      2                                unum      : unitnum);
1265:D      2 var
1266:D      -16 2     volpass      : name_type;
1267:D      -18 2     path        : path_start_type;
1268:D      -22 2     nfns        : integer;
1269:D      -130 2     nsa         : name_set_array_three;
1270:C      2     begin
1271:C      2         with f do
1272:C      3         if (strlen(ftid) = 0) and not fisnew then
1273:C      4         setioreresult(ord(ibadtitle))
1274:C      4         else
1275:C      4         begin
1276:C      4             if fmodified then
1277:C      5             if not (flockable and not flocked) then
1278:C      6             begin
1279:C      6                 seteofpack(funit,fileid,false,fleof);
1280:C      6                 if ioreresult = ord(ilostfile) then
1281:C      7                 fileid := -1;
1282:C      4             end;
1283:C      4             if fisnew and (not fanonymous) and (not fnosrmtemp) then
1284:C      5             srm_close_fileid(unum,foldfileid);
1285:C      4             srm_close_fileid(unum,fileid);
1286:C      4             setup_fns3(f,nsa);
1287:C      4             if ioreresult <> ord(ibadtitle) then
1288:C      5             begin
1289:C      5                 if (fisnew) and (not fnosrmtemp) then
1290:C      6                 begin
1291:C      6                     path      := start_root;
1292:C      6                     nfns       := 3;
1293:C      6                 end
1294:C      6                 else
1295:C      6                 begin
1296:C      6                     path      := start_alternate;
1297:C      6                     nfns       := 1;
1298:C      6                 end;
1299:C      5                 strttopacc16(fvid,volpass);
1300:C      5                 if fisnew then
1301:C      6                 nsa[nfns].password := temp_file_pass;
1302:C      5                 purgepack(unum,nfns,addr(nsa),path,pathid,volpass);
1303:C      5                 end;
1304:C      4                 if not (fisnew and fanonymous) then
1305:C      5                 srm_close_pathid(unum,pathid,fsavepathid);
1306:C      4                 end;
1307:C      2             end;
1308:S
1309:D      -6 1 (*****
1310:D      1 procedure srm_stretch(anyvar f      : fib;
1311:D      2                                unum      : unitnum);
1312:D      2 var
1313:D      -16 2     volpass      : name_type;
1314:D      -20 2     neweof       : integer;
1315:C      2     begin
1316:C      2         with f do
1317:C      3         begin
1318:C      3             neweof       := ((fpos div extentsize) + 1) * extentsize;
```



```

1319:C      3      seteofpack(funit,fileid,false,neweof);
1320:C      3      if ioresult = ord(inoerror) then
1321:C      4      begin
1322:C      4          fpecf      := neweof;
1323:C      4          fmodified  := true;
1324:C      4      end;
1325:C      3      end;
1326:C      2      end;
1327:C      1
1328:D      -6 1 (*****
1329:D      1 procedure srm_close_file(anyvar f      : fib;
1330:D      2                                unum    : unitnum);
1331:D      2
1332:D      2 var
1333:D      2     volpass      : name_type;
1334:D      2     nsal         : name_set_array_three;
1335:D      2     fns          : file_name_set;
1336:D      2     tempioresult : integer;
1337:D      2     ext1         : integer;
1338:D      2     saveeof      : integer;
1339:D      2     savefileid   : file_id_type;
1340:D      2     useleft      : gang_file_codes;
1341:D      2     pcs          : protect_code_set;
1342:C      2 begin
1343:C      2     with f do
1344:C      3     begin
1345:C      3         if fmodified then
1346:C      4         begin
1347:C      4             seteofpack(funit,fileid,false,fleof);
1348:C      4             if ioresult = ord(ilostfile) then
1349:C      5                 fileid := -1;
1350:C      4         end;
1351:C      3         if not fisnew then
1352:C      4             srm_close_fileid(unum,fileid)
1353:C      4         else
1354:C      5             if ioresult <> ord(inoerror) then
1355:C      6                 srm_purge_file(f,unum)
1356:C      5             else
1357:C      6                 begin
1358:C      6                     strtapaoc16(fvid,volpass);
1359:C      6                     setup_fns3(f,nsal);
1360:C      6                     setup_fns(f,fns);
1361:C      6                     if (ioresult = ord(inoerror)) and (not fnosrmtemp) then
1362:C      7                         if not foverwritten then
1363:C      8                             begin
1364:C      8                                 if not fanonymous then
1365:C      9                                     begin
1366:C      9                                         srm_close_fileid(unum,foldfileid);
1367:C      9                                         purgepack(unum,1,addr(fns),start_alternate,pathid,volpass);
1368:C      9                                     end;
1369:C      8                                 if ioresult = ord(inofile) then
1370:C      9                                     ioresult := ord(inoerror);
1371:C      8                                 if ioresult <> ord(inoerror) then
1372:C      9                                     srm_purge_file(f,unum);
1373:C      8                                 end
1374:C      7                             else
1375:C      8                                 begin
1376:C      8                                     if foldfileid < 0 then
1377:C      9                                         foverwritten := false
1378:C      8                                     else

```

```

1379:C      6      begin
1380:C      6          exchangeback(unum,foldfileid,fileid);
1381:C      6          srm_close_fileid(unum,foldfileid);
1382:C      6          srm_close_pathid(unum,pathid,fsavepathid);
1383:C      6      end;
1384:C      7      srm_purge_file(f,unum);
1385:C      5      end;
1386:C      5      if (ioresult = ord(inoerror)) and (not foverwritten) then
1387:C      6      if fext1 <> SYSTMTYPE then (not SYSTM file)
1388:C      7      begin
1389:C      7          srm_close_fileid(unum,fileid);
1390:C      7          if (ioresult = ord(inoerror)) and (not fnosrmtemp) then
1391:C      8              createlinkpack(unum,0,addr(nsal),start_root,pathid,volpass,
1392:C      8                          1,addr(fns),start_alternate,pathid,volpass,true);
1393:C      7          if ioresult <> ord(inoerror) then
1394:C      8              srm_purge_file(f,unum)
1395:C      7          else
1396:C      8              begin
1397:C      8                  fns.password := temp_file_pass;
1398:C      8                  with pcs do
1399:C      9                  begin
1400:C      9                      password := temp_file_pass;
1401:C      9                      capabilities := nocapabilities;
1402:C      9                  end;
1403:C      8                  tempioresult := ioresult;
1404:C      8                  changeprotectpack(unum,1,addr(fns),start_alternate,
1405:C      8                                  pathid,volpass,1,addr(pcs));
1406:C      8                  ioresult := tempioresult;
1407:C      8              end;
1408:C      7          end;
1409:C      7          else
1410:C      7              begin
1411:C      7                  (SYSTM file)
1412:C      7                  savefileid := fileid;
1413:C      7                  saveeof := fleof;
1414:C      7                  fpos := saveeof;
1415:C      7                  if fpos > 0 then
1416:C      8                      ext1 := fpos
1417:C      8                  else
1418:C      7                      ext1 := extentsize;
1419:C      7                      useleft.1 := fext1;
1420:C      7                      fns.password := ' ';
1421:C      7                      createpack(unum,1,addr(fns),start_alternate,pathid,
1422:C      7                                  volpass,0,nil,useleft,data_records,
1423:C      7                                  256,ext1,extentsize,fstartaddress);
1424:C      8                      if ioresult = ord(inoerror) then
1425:C      9                          doopenpack(unum,f,1,fns,start_alternate,volpass,exclusive_share_code,false);
1426:C      8                      if ioresult = ord(inoerror) then
1427:C      9                          copypack(unum,savefileid,0,fileid,0,saveeof);
1428:C      8                          srm_close_fileid(unum,fileid);
1429:C      8                          srm_close_pathid(unum,pathid,fsavepathid);
1430:C      8                          fileid := savefileid;
1431:C      7                          srm_purge_file(f,unum);
1432:C      7                      end;
1433:C      5                  end;
1434:C      3                  srm_close_pathid(unum,pathid,fsavepathid);
1435:C      3                  end;
1436:C      2      end;
1437:D      -6 1 (*****
1438:D      -6 1 procedure srm_get_vol_date(anyvar f      : datatimerec;

```



```
1439:D      2      unum : unitnum);
1440:D      2      type
1441:D      2      fibptr   = ^fib;
1442:D      2      var
1443:D      2      tempfibspace : packed array [1..sizeof(fib,0)] of char;
1444:D      -150
1445:C      2      begin
1446:C      3      with fibptr(addr(tempfibspace))^, packet_ptr.rfileinfo^.file_info do
1447:C      3      begin
1448:C      3          funit := unum;
1449:C      3          pathid := -1;
1450:C      3          fileid := -1;
1451:C      3          fpos := 0;
1452:C      3          fkind := datafile;
1453:C      3          left := DATATYPE;
1454:C      3          fisnew := true;
1455:C      3          fanonymous := true;
1456:C      3          fmodified := false;
1457:C      3          fopstring := nil;
1458:C      3          fnosrmtemp := false;
1459:C      3          setstrlen(fid,0);
1460:C      3          srm_create_file(fibptr(addr(tempfibspace))^,unum);
1461:C      3          if ioresult = ord(inoerror) then
1462:C      4          begin
1463:C      4              fileinfopack(unum,fileid);
1464:C      4              if ioresult = ord(inoerror) then
1465:C      5              with f do
1466:C      5                  translatedate(creation date,date,time);
1467:C      4              srm_purge_file(fibptr(addr(tempfibspace))^,unum);
1468:C      4              end;
1469:C      3          end;
1470:C      2      end;
1471:S      -6 1 (*****
1472:D      1 procedure srm_lock_file(anyvar f : fib;
1473:D      2      unum : unitnum);
1474:D      2      begin
1475:D      3      with f, packet_ptr.rlock^ do
1476:D      3      begin
1477:D      3          lockpack(unum,fileid,fwaitonlock);
1478:D      3          if ioresult = ord(inoerror) then
1479:D      4          if not success.value then
1480:D      4              setioresult(ord(ifilelocked))
1481:D      4          else
1482:D      5          begin
1483:D      5              fileinfopack(unum,fileid);
1484:D      5              if ioresult = ord(inoerror) then
1485:D      6              with packet_ptr.rfileinfo^.file_info do
1486:D      6              begin
1487:D      7                  fpeof := logical_eof;
1488:D      7                  fleof := logical_eof;
1489:D      7                  flocked := true;
1490:D      7                  end;
1491:D      5          end;
1492:D      3          end;
1493:D      2      end;
1494:S      -6 1 (*****
1495:D      1 procedure srm_unlock_file(anyvar f : fib;
1496:D      2      unum : unitnum);
1497:D      2      begin
```

```
1499:C      2      with f do
1500:C      3      begin
1501:C      3          if ioresult = ord(inoerror) then
1502:C      4          begin
1503:C      4              call(am,addr(f),flush,f,0,0);
1504:C      4              flastpos := -1;
1505:C      4              if ioresult = ord(inoerror) then
1506:C      5              begin
1507:C      5                  if fmodified then
1508:C      6                  seteofpack(unum,fileid,false,fleof);
1509:C      5                  if ioresult = ord(inoerror) then
1510:C      6                  unlockpack(unum,fileid);
1511:C      5                  if ioresult = ord(inoerror) then
1512:C      6                  flocked := false;
1513:C      5                  end;
1514:C      4          end;
1515:C      3          end;
1516:C      2      end;
1517:S      -6 1 (*****
1518:D      1 procedure srm_stripc(anyvar f : fib);
1519:D      2      var
1520:D      2      s : string[255];
1521:D      2      findx : integer;
1522:D      2      sindx : integer;
1523:D      2      namelen : integer;
1524:D      2      passlen : integer;
1525:D      2      i : integer;
1526:D      2      ch : char;
1527:D      2      skip : boolean;
1528:D      2      inpass : boolean;
1529:D      2      nopathyet : boolean;
1530:D      2      begin
1531:D      3      namelen := 0;
1532:D      3      passlen := 0;
1533:D      3      findx := 1;
1534:D      3      sindx := 0;
1535:D      3      setstrlen(s,255);
1536:D      3      inpass := false;
1537:D      3      nopathyet := true;
1538:D      3      with f do
1539:D      3      begin
1540:D      4          if ftitle[1] = '<' then (skip over volume password)
1541:D      4          repeat
1542:D      5              findx := findx + 1;
1543:D      5              if (findx > name_type_len + 3) or (findx > strlen(ftitle)) then
1544:D      6              setioresult(ord(ibadpass));
1545:D      5              until (ftitle[findx-1] = '>') or (ioresult <> ord(inoerror));
1546:D      5              while (findx <= strlen(ftitle)) and (ioresult = ord(inoerror)) do
1547:D      6              begin
1548:D      7                  skip := false;
1549:D      7                  ch := ftitle[findx];
1550:D      7                  if inpass then
1551:D      8                  begin
1552:D      8                      skip := true;
1553:D      8                      if ch = '>' then
1554:D      9                      inpass := false
1555:D      8                  end
1556:D      7                  else
```



```

1559:C      6      passlen := passlen + 1;
1560:C      5      end
1561:C      5      else if ch = '/' then
1562:C      6      begin
1563:C      6      nopathyet := true;
1564:C      6      inpass := false;
1565:C      6      namelen := 0;
1566:C      6      passlen := 0;
1567:C      6      end
1568:C      6      else if ch = '<' then
1569:C      7      if nopathyet then
1570:C      8      begin
1571:C      8      nopathyet := false;
1572:C      8      inpass := true;
1573:C      8      skip := true;
1574:C      8      end;
1575:S      4
1576:C      4      if not skip then
1577:C      5      begin
1578:C      5      if ch = '/' then
1579:C      6      begin
1580:C      6      if s[sindx] = '/' then
1581:C      7      setioresult(ord(ibadtitle));
1582:C      6      end
1583:C      6      else
1584:C      6      namelen := namelen + 1;
1585:C      6      s[sindx] := s[sindx + 1];
1586:C      6      s[sindx] := ch;
1587:C      6      end;
1588:S      4
1589:C      4      findx := findx + 1;
1590:S      4
1591:C      4      if (namelen > name_type_len) then
1592:C      5      setioresult(ord(ibadtitle));
1593:C      5      else if (passlen > name_type_len) then
1594:C      6      setioresult(ord(ibadpass));
1595:C      4      end;
1596:C      3
1597:C      3      if ioreresult = ord(inoerror) then
1598:C      4      begin
1599:C      4      setstrlen(s,sindx);
1600:C      4      i := 0;
1601:C      4      while [s[sindx-i] <> '/'] and (i < sindx) do
1602:C      4      i := i + 1;
1603:C      4      if i = 0 then
1604:C      5      setioresult(ord(ibadtitle));
1605:C      5      else
1606:C      5      begin
1607:C      5      setstrlen(ftid,0);
1608:C      5      strmove(i,s,sindx-i+1,ftid,1);
1609:C      5      setstrlen(ftitle,0);
1610:C      5      strmove(sindx-1,s,1,ftitle,1);
1611:C      5      end;
1612:C      4      end;
1613:C      3      end;
1614:C      2      end;
1615:S      2
1616:D      -6 1 (*****
1617:D      1 procedure srmdaminit;
1618:C      2 begin

```

```

1619:C      2      srm_init;
***WARNING: (line 1620): 'ADDR' of a constant may not be supported on other implementations
1620:C      2      passwordarrayptr := addr(constpasswordarray);
1621:C      2      end;
1622:S      2
1623:D      -6 1 (*****
1624:D      1 procedure srmdam(anyvar f : fib;
1625:D      2      unum : unitnum;
1626:D      2      request : damrequesttype);
1627:D      2      var
1628:D      2      holdpathid : integer;
1629:D      2      savepathid : integer;
1630:D      2      savefileid : integer;
1631:D      2      saveftid : tid;
1632:D      2      saveffpw : passtype;
1633:D      2      savefvid : vid;
1634:D      2      saveftitle : tid;
1635:D      2      savefsavepathid : boolean;
1636:D      2      fisafib : boolean;
1637:C      2      begin
1638:C      2      ioreresult := ord(inoerror);
1639:C      2      srmsavesc := 0;
1640:C      2      lockup;
1641:C      2      fisafib := false;
1642:C      2      try
1643:C      3      with f, unitable[unum] do
1644:C      4      if offline then
1645:C      5      ioreresult := ord(znodevice);
1646:C      5      else
1647:C      6      begin
1648:C      6      if request in [opendirectory,
1649:C      6      openparentdir,
1650:C      6      closedirectory,
1651:C      6      catalog,
1652:C      6      catpasswords,
1653:C      6      setpasswords,
1654:C      6      openfile,
1655:C      6      createfile,
1656:C      6      overwritefile,
1657:C      6      makedirectory,
1658:C      6      closefile,
1659:C      6      changename,
1660:C      6      duplicatelink,
1661:C      6      purgename,
1662:C      6      lockfile,
1663:C      6      unlockfile,
1664:C      6      purgefile,
1665:C      6      setunitprefix,
1666:C      6      stretchit ] then (f is a fib)
1667:C      6      begin
1668:C      6      fisafib := true;
1669:C      6      if strlen(ftid) > tidlength then (fix uninitialized fib strings)
1670:C      6      setstrlen(ftid,0);
1671:C      6      if strlen(ffpw) > passlength then
1672:C      6      setstrlen(ffpw,0);
1673:C      6      if strlen(fvid) > vidlength then
1674:C      6      setstrlen(fvid,0);
1675:C      6      savepathid := pathid; (save fib fields to be restored on error)
1676:C      6      savefileid := fileid;
1677:C      6

```



```

1678:C      6      savefsavepathid := fsavepathid;
1679:C      6      saveftid      := ftid;
1680:C      6      saveffpw      := ffpw;
1681:C      6      savefvid      := fvid;
1682:C      6      if strlen(ftitle) > fdieng then
1683:C      7      setstrlen(saveftitle,0)
1684:C      7      else
1685:C      7      saveftitle      := ftitle;
1686:C      6      end;
1687:C      5      case request of
1688:C      6      opendir,
1689:C      6      opendirparentdir : begin
1690:C      6      srm_open_dir(f,unum,open_directory,request = opendirparentdir);
1691:C      6      if ioresult = ord(inoerror) then
1692:C      7      srm_get_dir_info(fwindow^,pathid,unum,true,false);
1693:C      6      end;
1694:S
1695:C      6      closedirectory : begin
1696:C      6      fsavepathid      := false;
1697:C      6      srm_close_pathid(unum,pathid,false);
1698:C      6      end;
1699:S
1700:C      6      catalog      : srm_catalog(f,unum);
1701:S
1702:C      6      catpasswords : begin
1703:C      6      srm_open_dir(f,unum,open_directory,true);
1704:C      6      if ioresult = ord(inoerror) then
1705:C      7      begin
1706:C      7      srm_cat_pass(f,unum);
1707:C      7      srm_close_pathid(unum,pathid,fsavepathid);
1708:C      7      end;
1709:C      6      end;
1710:S
1711:C      6      setpasswords : begin
1712:C      6      srm_open_dir(f,unum,open_directory,true);
1713:C      6      if ioresult = ord(inoerror) then
1714:C      7      begin
1715:C      7      srm_set_pass(f,unum);
1716:C      7      srm_close_pathid(unum,pathid,fsavepathid);
1717:C      7      end;
1718:C      6      end;
1719:S
1720:C      6      openfile      : begin
1721:C      6      fisnew          := false;
1722:C      6      fnosrmtemp      := true;      (default case)
1723:C      6      srm_open_dir(f,unum,open_directory,true );
1724:C      6      if ioresult = ord(inoerror) then
1725:C      7      begin
1726:C      7      srm_open_file(f,unum);
1727:C      7      if ioresult <> ord(inoerror) then
1728:C      8      srm_close_pathid(unum,pathid,fsavepathid);
1729:C      7      end;
1730:C      6      end;
1731:S
1732:C      6      createfile,
1733:C      6      overwritefile : begin
1734:C      6      fisnew          := true;
1735:C      6      fnosrmtemp      := false;
1736:C      6      foverwritten    := request = overwritefile;
1737:C      6      if not fanonymous then

```

```

1738:C      7      srm_open_dir(f,unum,open_directory,true );
1739:C      6      if ioresult = ord(inoerror) then
1740:C      7      begin
1741:C      7      srm_create_file(f,unum);
1742:C      7      if ioresult <> ord(inoerror) then
1743:C      8      if not fanonymous then
1744:C      9      srm_close_pathid(unum,pathid,fsavepathid);
1745:C      7      end;
1746:C      6      end;
1747:S
1748:C      6      makedirectory : begin
1749:C      6      holdpathid      := pathid;
1750:C      6      srm_open_dir(f,unum,open_directory,false);
1751:C      6      if ioresult = ord(inoerror) then
1752:C      7      begin
1753:C      7      srm_create_dir (f,unum);
1754:C      7      srm_close_pathid(unum,pathid,fsavepathid);
1755:C      7      end;
1756:C      6      pathid      := holdpathid;
1757:C      6      end;
1758:S
1759:C      6      closefile      : if (fisnew and fanonymous) then
1760:C      7      srm_purge_file(f,unum)
1761:C      7      else
1762:C      7      srm_close_file(f,unum);
1763:S
1764:C      6      changename      : srm_change_name(f,unum);
1765:S
1766:C      6      duplicatelink : srm_dup_link(f,unum);
1767:S
1768:C      6      purgename      : begin
1769:C      6      holdpathid      := pathid;
1770:C      6      srm_open_dir(f,unum,open_directory,true);
1771:C      6      if ioresult = ord(inoerror) then
1772:C      7      begin
1773:C      7      srm_purge_name(f,unum);
1774:C      7      srm_close_pathid(unum,pathid,fsavepathid);
1775:C      7      end;
1776:C      6      pathid      := holdpathid;
1777:C      6      end;
1778:S
1779:C      6      lockfile      : srm_lock_file(f,unum);
1780:S
1781:C      6      unlockfile      : srm_unlock_file(f,unum);
1782:S
1783:C      6      purgefile      : srm_purge_file(f,unum);
1784:S
1785:C      6      setunitprefix : srm_set_unit_prefix(f,unum);
1786:S
1787:C      6      stretchit      : srm_stretch(f,unum);
1788:S
1789:C      6      getvolumename : srm_get_vol_name(f,unum);
1790:S
1791:C      6      getvolumedate : srm_get_vol_date(f,unum);
1792:S
1793:C      6      setvolumedate,
1794:C      6      crunch          : (do nothing, but no error);
1795:S
1796:C      6      stripname      : srm_strip(f);
1797:S

```



```
1798:C      6      otherwise      setioresult(ord(ibadrequest));
1799:C      6      end;
1800:S
1801:C      5      if (ioresult <> ord(inoerror)) and fisafib then (restore fib for subsequent calls)
1802:C      6      begin
1803:C      6          pathid      := savepathid;
1804:C      6          fileid       := savefileid;
1805:C      6          fsavepathid   := savefsavepathid;
1806:C      6          ftid         := saveftid;
1807:C      6          ffpw         := saveffpw;
1808:C      6          fvid         := savefvid;
1809:C      6          if strlen(saveftitle) > 0 then
1810:C      7              ftitle      := saveftitle;
1811:C      6          end;
1812:C      5          if ioresult = ord(isrmcatchall) then
1813:C      6              if srmsavesc <> 0 then
1814:C      7                  escape(srmsavesc);
1815:C      5          end;
1816:C      3      recover
1817:C      3      begin
1818:C      3          if escapecode = ioescapecode then
1819:C      4              setioresult(ord(isrmcatchall))
1820:C      4          else
1821:C      4              begin
1822:C      4                  lockdown;
1823:C      4                  escape(escapecode);
1824:C      4              end;
1825:C      3          end;
1826:C      2      lockdown;
1827:C      2      end;
1828:S
1829:C      1      end; (srmdammodule)
1830:S
1831:D      1      import
1832:D      1      srmdammodule;
1833:C      1      begin (program init_srm)
1834:C      1      srmdaminit;
1835:C      1      end.
1836:S
```

No errors. 3 warnings.

***** Nonstandard language features enabled *****

SRM_DRV

Description

SRM_DRV is the low-level support for SRMAM and SRMDAM. Primarily, it serves as an SRM packet constructor, sender and receiver.

Requirements

The I/O library drivers for the 98629 card (DATA_COMM).

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:      (*
2:      (c) Copyright Hewlett-Packard Company, 1983.
3:      All rights are reserved. Copying or other
4:      reproduction of this program except for archival
5:      purposes is prohibited without the prior
6:      written consent of Hewlett-Packard Company.
7:
8:
9:
10:
11:      RESTRICTED RIGHTS LEGEND
12:
13:      Use, duplication, or disclosure by the Government
14:      is subject to restrictions as set forth in
15:      paragraph (b) (3) (B) of the Rights in Technical
16:      Data and Computer Software clause in
17:      DAR 7-104.9(a).
18:
19:      HEWLETT-PACKARD COMPANY
20:      0 Fort Collins, Colorado      *)
21:
22:      0 $copyright 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'S
23:      0 $range off$
24:      0 $debug off$
25:      0 $modcal$
26:      0 $sysprog$
27:      0 $ucsd$
28:      0 module srm;
29:      1 $search 'IOL18:KERNEL'S
30:      1 import iodeclarations,
31:      1     sysglobals,
32:      1     loader;
33:      1 export
34:
35:      1 $include 'INIT:SRM_TYPES'S
36:      1 type
37:      {
38:      byte      = 0..255;
39:      halfword  = -32768..32767;
40:
41:      1 $include 'PACKETS'S
42:      {
43:      *-----*
44:      *      G A N G L I A   M E S S A G E   F O R M A T S
45:      *      A N D   R E L A T E D   D E F I N I T I O N S
46:      *-----*
47:      }
48:      1
49:      1
50:      1 CONST
51:      1
52:      1     name_type_len = 16;
53:
54:      {
55:      | Message Packet Sizes For Messages That Are Sent To Ganglia
56:      |
57:      size_to_are_you_alive = 12;
58:      size_to_cat = 128;      (* plus 36 * # file name sets *)

```

```

59:      1     size_to_catprotect = 128;      (* plus 36 * # file name sets *)
60:      1     size_to_changeprotect = 116;      (* plus (36 * # file name sets) +
61:      1                                     (24 * # file name sets) *)
62:      1     size_to_change_vol_label= 132;
63:      1     size_to_close = 56;
64:      1     size_to_copy = 32;
65:      1     size_to_create = 152;      (* plus (36 * # file name sets) +
66:      1                                     (24 * # protect code sets) *)
67:      1     size_to_createlink = 144;      (* plus 36 * # file name sets *)
68:      1     size_to_flock = 20;
69:      1     size_to_funlock = 20;
70:      1     size_to_gang_cleanup = 16;
71:      1     size_to_info = 20;
72:      1     size_to_init_vol = 172;      (* plus 24 * # protect code sets *)
73:      1     size_to_open = 132;      (* plus 36 * # file name sets *)
74:      1     size_to_position = 28;
75:      1     size_to_purge = 112;      (* plus 36 * # file name sets *)
76:      1     size_to_read = 40;
77:      1     size_to_set_eof = 44;
78:      1     size_to_set_date = 44;
79:      1     size_to_volStatus = 84;
80:      1     size_to_write = 48;      (* plus the size of the record <= 512 *)
81:      1     size_to_xchg_data = 140;      (* plus 36 * # file name sets *)
82:      1     size_to_xchg_open = 20;
83:      1
84:      1
85:      {
86:      | Message Packet Sizes For Messages Ganglia Will Send Back
87:      |
88:      size_from_are_you_alive = 16;
89:      1     size_from_cat = 24;      (* plus 68 * # entries returned *)
90:      1     size_from_catprotect = 24;      (* plus 20 * # prot code entries returned *)
91:      1     size_from_changeprotect = 16;
92:      1     size_from_change_vol_label=16;
93:      1     size_from_close = 16;
94:      1     size_from_copy = 20;
95:      1     size_from_create = 16;
96:      1     size_from_createlink = 16;
97:      1     size_from_flock = 20;
98:      1     size_from_funlock = 16;
99:      1     size_from_gang_cleanup = 16;
100:      1     size_from_gang_error = 16;
101:      1     size_from_info = 88;
102:      1     size_from_init_vol = 16;
103:      1     size_from_open = 52;
104:      1     size_from_position = 16;
105:      1     size_from_purge = 16;
106:      1     size_from_read = 36;      (* plus the size of the record <= 512 *)
107:      1     size_from_set_eof = 16;
108:      1     size_from_set_date = 16;
109:      1     size_from_volStatus = 40;
110:      1     size_from_write = 20;
111:      1     size_from_xchg_data = 16;
112:      1     size_from_xchg_open = 16;
113:      1
114:      1
115:      {
116:      | Ganglia Message Request Types, Modes, And Codes
117:      |
118:      1

```



```

119:D 1
120:D 1
121:D 1
122:D 1
123:D 1
124:D 1
125:D 1
126:D 1
127:D 1
128:D 1
129:D 1
130:D 1
131:D 1
132:D 1
133:D 1
134:D 1
135:D 1
136:D 1
137:D 1
138:D 1
139:D 1
140:D 1
141:D 1
142:S 1
143:D 1
144:D 1
145:D 1
146:D 1
147:D 1
148:D 1
149:D 1
150:D 1
151:S 1
152:D 1
153:D 1
154:D 1
155:D 1
156:D 1
157:D 1
158:D 1
159:D 1
160:D 1
161:D 1
162:D 1
163:D 1
164:D 1
165:D 1
166:S 1
167:D 1
168:D 1
169:D 1
170:D 1
171:D 1
172:D 1
173:D 1
174:D 1
175:S 1
176:S 1
177:S 1
178:D 1

```

```

(***) File Access Method Requests (***)
req_oldread = 0; {Old read packet}
req_write = 1; {Write a record to an open file.}
req_position = 2; {Position an open file.}
req_read = 3; {Read a record from an open file.}
req_set_eof = 4;

(***) Sharing Requests (***)
req_flock = 8; {Lock an open file for exclusive use.}
req_funlock = 9; {Unlock an open file.}
req_info = 10;
req_close = 13; {Close an open file.}

(***) Directory Access Requests (***)
req_open = 14;
req_purgelink = 15;
req_catalog = 16; {List all the files in a directory.}
req_create = 17;
req_createlink = 18;
req_changeprotect = 19; {Change the passwords on a file.}
req_catprotect = 20; {List all the passwords on a file.}
req_xchg_data = 21;

(***) Volume Access Requests (***)
req_volstatus = 22;
req_init_vol = 23;
req_label = 24;

(***) File Transfer Requests (***)
req_xchg_open = 29;
req_copy = 30;

(***) Ganglia Requests (***)
req_gang_cleanup = 1000;
req_are_you_alive = 1001;

(***) Record Modes (***)
data_records = 0;
directory_records = 1;

(***) Share Codes (***)
exclusive_share_code = 0;
shared_share_code = 1;
closed_share_code = 2;
corrupt_share_code = 3;

(***) Access Codes (***)
random_access = 0;
sequential_access = 1;
primitive_access = 2;

```

TYPE

```

{
Enumerated Types And Minor Structures Used To Make Up More
Major Structures Described In The Next Section
}

```

```

179:D 1
180:D 1
181:D 1
182:D 1
183:D 1
184:D 1
185:D 1
186:D 1
187:D 1
188:D 1
189:D 1
190:D 1
191:D 1
192:D 1
193:D 1
194:D 1
195:D 1
196:D 1
197:D 1
198:D 1
199:D 1
200:D 1
201:D 1
202:D 1
203:D 1
204:D 1
205:D 1
206:D 1
207:D 1
208:D 1
209:D 1
210:D 1
211:S 1
212:D 1
213:D 1
214:D 1
215:D 1
216:D 1
217:D 1
218:D 1
219:D 1
220:D 1
221:S 1
222:D 1
223:D 1
224:D 1
225:D 1
226:D 1
227:D 1
228:D 1
229:D 1
230:D 1
231:D 1
232:D 1
233:D 1
234:D 1
235:D 1
236:D 1
237:D 1
238:D 1

```

```

access_code_type =
(
ac_manager,
ac_read,
ac_write,
ac_search,
ac_purgelink,
ac_createlink,
ac_execute,
ac_generic1,ac_generic09,ac_generic17,
ac_generic2,ac_generic10,ac_generic18,
ac_generic3,ac_generic11,ac_generic19,
ac_generic4,ac_generic12,ac_generic20,
ac_generic5,ac_generic13,ac_generic21,
ac_generic6,ac_generic14,ac_generic22,
ac_generic7,ac_generic15,ac_generic23,
ac_generic8,ac_generic16,ac_generic24,
ac_generic25);

file_id_type = integer;

(***) Filler Definitions Used In Padding Out Message Packet Layouts (***)
gang_boolean_filler = boolean;
gang_32bit_filler = integer;
gang_16bit_filler = -32768..32767;

long_boolean = record case integer of
1: (i: integer);
2: (fill1: gang_16bit_filler;
fill2: gang_boolean_filler;
value: boolean);
end;

gang_file_codes = packed record
case integer of
0: (i: integer);
1: (s1: shortint;
s2: shortint);
2: (firstword: -32768..32767;
division_code: 0..63;
dcd_system_type: 0..15;
dcd_file_type: 0..63);
end;

gang_open_type =
(
open_data, {Normal data file}
open_protected_directory, {Directory not closed on cleanup}
open_directory {Directory closed on cleanup}
);

name_type = packed array [1..name_type_len] of char;

path_start_type =
(
start_root, {Start search at root directory}
start_alternate {Start search at a working directory.}
);

position_type =
(pos_absolute, {Position at absolute location in file}

```



```

239:D      1      pos_seq      (Position relative to current record)
240:D      1      );
241:D      1
242:D      1
243:S      1      {-----+
244:S      1      | Data Structures Which Are Part Of Message Packet Layouts
245:D      1      +-----+}
246:D      1
247:D      1      access_capabilities = packed array [ access_code_type ] of boolean;
248:D      1
249:D      1      date_type = record
250:D      1      fill: gang_16bit_filler;
251:D      1      date: packed record
252:D      1      month: 0..12;
253:D      1      day: 0..31;
254:D      1      year: 0..100;
255:D      1      end;
256:D      1      seconds_since_midnight : integer
257:D      1      end;
258:D      1
259:D      1      device_address_type = record
260:D      1      address1 : integer;
261:D      1      haddress : integer;
262:D      1      unit_num : integer;
263:D      1      volume_num : integer
264:D      1      end;
265:D      1
266:D      1      file_header_type = record
267:D      1      num_file_name_sets : integer;
268:D      1      working_directory : file_id_type;
269:D      1      filler1 : gang_16bit_filler;
270:D      1      path_type : path_start_type;
271:D      1      root_password : name_type;
272:D      1      end;
273:D      1
274:D      1      file_info_type = record
275:D      1      file_name : name_type;
276:D      1      open_flag : long_boolean;
277:D      1      snr_code : integer;
278:D      1      file_code : gang_file_codes;
279:D      1      record_mode : integer;
280:D      1      max_record_size : integer;
281:D      1      max_file_size : integer;
282:D      1      creation_date : date_type;
283:D      1      last_access_date : date_type;
284:D      1      capabilities : access_capabilities;
285:D      1      logical_eof : integer;
286:D      1      physical_size : integer;
287:D      1      end;
288:S      1
289:D      1      file_name_set = record
290:D      1      file_name : name_type;
291:D      1      password : name_type;
292:D      1      filler : gang_32bit_filler
293:D      1      end;
294:D      1
295:D      1      linkfillertype = packed record
296:D      1      requeue: boolean;
297:D      1      zit1: byte; { unused }
298:D      1      zit2: integer; { unused }

```

```

299:D      1      oddbytefiller: byte; { unused }
300:D      1      dstaddr: byte;
301:D      1      sourceaddr: byte; { These three bytes }
302:D      1      len_lobyte: byte; { are filled in by }
303:D      1      len_hibyte: byte; { the transmitting card }
304:D      1      level: byte;
305:D      1      end;
306:D      1
307:D      1      msg_packet_type = packed array [1..800 (or so) ] of char;
308:D      1      msg_packet_ptr = ^msg_packet_type;
309:D      1
310:D      1      owner_id_type = record
311:D      1      id: integer;
312:D      1      end;
313:D      1
314:D      1      protect_code_set = record
315:D      1      capabilities : access_capabilities;
316:D      1      password : name_type;
317:D      1      filler : gang_32bit_filler
318:D      1      end;
319:D      1
320:D      1      return_header_type = record
321:D      1      linkfiller : linkfillertype;
322:D      1      message_length : integer;
323:D      1      return_request_type : integer;
324:D      1      user_sequencing_field : integer;
325:D      1      status : integer
326:D      1      end;
327:D      1
328:D      1      send_header_type = record
329:D      1      linkfiller : linkfillertype;
330:D      1      message_length : integer;
331:D      1      send_request_type : integer;
332:D      1      user_sequencing_field : integer
333:D      1      end;
334:D      1
335:D      1      volume_header_type = record
336:D      1      filler1 : gang_32bit_filler;
337:D      1      driver_name : name_type;
338:D      1      catalogue_organization : name_type;
339:D      1      device_address_present : long_boolean;
340:D      1      device_address : device_address_type;
341:D      1      volume_name : name_type;
342:D      1      end;
343:D      1
344:D      1      volume_info_type = record
345:D      1      free_blocks : integer;
346:D      1      bad_blocks : integer;
347:D      1      media_origin : integer;
348:D      1      interleave : integer;
349:D      1      volume_label : name_type;
350:D      1      end;
351:D      1
352:D      1
353:D      1
354:S      1      {-----+
355:S      1      | Send and return record layouts for request: Are You Alive ?!
356:D      1      +-----+}
357:D      1
358:D      1      send_are_you_alive = record

```



```

359:D      1      send_mess_header : send_header_type;
360:D      1      end;
361:D      1
362:D      1      return_are_you_alive = record
363:D      1      return_mess_header : return_header_type;
364:D      1      end;
365:D      1
366:D      1
367:S      1      (+-----+
368:S      1      | Send and return record layouts for request: Catalogue |
369:D      1      +-----+)
370:D      1
371:D      1      send_catalogue = record
372:D      1      send_mess_header : send_header_type;
373:D      1      max_num_files : integer;
374:D      1      file_index : integer;
375:D      1      filler1 : gang_32bit_filler; {actual_num_files}
376:D      1      volume_name_header : volume_header_type;
377:D      1      file_name_header : file_header_type;
378:D      1      filler2 : gang_32bit_filler;
379:D      1      start_name_sets : integer;
380:D      1      end;
381:D      1
382:D      1      return_catalogue = record
383:D      1      return_mess_header : return_header_type;
384:D      1      filler1 : gang_32bit_filler; {file_index}
385:D      1      actual_num_files : integer;
386:D      1      cat_info : array [1..7] of file_info_type
387:D      1      end;
388:D      1
389:D      1
390:S      1      (+-----+
391:S      1      | Send and return record layouts for request: Catalogue Protect Codes |
392:D      1      +-----+)
393:D      1
394:D      1      send_cat_passwords = record
395:D      1      send_mess_header : send_header_type;
396:D      1      max_num_passwords : integer;
397:D      1      filler1 : gang_32bit_filler; {actual_passwords}
398:D      1      password_index : integer;
399:D      1      volume_name_header : volume_header_type;
400:D      1      file_name_header : file_header_type;
401:D      1      filler2 : gang_32bit_filler;
402:D      1      start_name_sets : integer;
403:D      1      end;
404:D      1
405:D      1      return_cat_passwords = record
406:D      1      return_mess_header : return_header_type;
407:D      1      actual_num_passwords : integer;
408:D      1      filler1 : gang_32bit_filler;
409:D      1      password_info : array [1..24] of record
410:D      1      password : name_type;
411:D      1      capabilities : access_capabilities;
412:D      1      end;
413:D      1      end;
414:D      1
415:D      1
416:S      1      (+-----+
417:S      1      | Send and return record layouts for request: Change Protect Codes |
418:D      1      +-----+)

```

```

419:D      1
420:D      1      send_change_protect_codes = record
421:D      1      send_mess_header : send_header_type;
422:D      1      volume_name_header : volume_header_type;
423:D      1      file_name_header : file_header_type;
424:D      1      num_protect_code_sets : integer;
425:D      1      start_name_sets : integer;
426:D      1      end;
427:D      1
428:D      1      return_change_protect_codes = record
429:D      1      return_mess_header : return_header_type;
430:D      1      end;
431:D      1
432:D      1
433:S      1      (+-----+
434:S      1      | Send and return record layouts for request: Change Volume Label |
435:D      1      +-----+)
436:D      1
437:D      1      send_change_volume_label = record
438:D      1      send_mess_header : send_header_type;
439:D      1      volume_name_header : volume_header_type;
440:D      1      password : name_type;
441:D      1      new_volume_name : name_type;
442:D      1      new_vol_password : name_type;
443:D      1      end;
444:D      1
445:D      1      return_change_volume_label = record
446:D      1      return_mess_header : return_header_type;
447:D      1      end;
448:D      1
449:D      1
450:S      1      (+-----+
451:S      1      | Send and return record layouts for request: Close A File |
452:D      1      +-----+)
453:D      1
454:D      1      send_closeafile = record
455:D      1      send_mess_header : send_header_type;
456:D      1      file_id : file_id_type;
457:D      1      directory_password : name_type;
458:D      1      file_password : name_type;
459:D      1      filler5 : long_boolean;
460:D      1      nodeallocate : long_boolean;
461:D      1      end;
462:D      1
463:D      1      return_closeafile = record
464:D      1      return_mess_header : return_header_type;
465:D      1      end;
466:D      1
467:D      1
468:S      1      (+-----+
469:S      1      | Send and return record layouts for request: Copy File To File |
470:D      1      +-----+)
471:D      1
472:D      1      send_copy = record
473:D      1      send_mess_header : send_header_type;
474:D      1      source_file_id : file_id_type;
475:D      1      source_offset : integer;
476:D      1      destination_file_id : file_id_type;
477:D      1      destination_offset : integer;
478:D      1      requested : integer;

```



```

479:D      1      end;
480:D      1
481:D      1      return_copy = record
482:D      1          return_mess_header      : return_header_type;
483:D      1          actual                      : integer;
484:D      1      end;
485:D      1
486:D      1
487:S      1      (+-----+
488:S      1      | Send and return record layouts for request: Create A File |
489:D      1      +-----+)
490:D      1
491:D      1      send_createfile = record
492:D      1          send_mess_header      : send_header_type;
493:D      1          volume_name_header    : volume_header_type;
494:D      1          file_name_header      : file_header_type;
495:D      1          file_code             : gang_file_codes;
496:D      1          record_mode           : integer;
497:D      1          max_record_size       : integer;
498:D      1          first_extent          : integer;      (Size in logical records.)
499:D      1          contiguous_first_extent : long_boolean;
500:D      1          secondary_extent      : integer;
501:D      1          max_file_size         : integer;
502:D      1          boot_start_address    : integer;
503:D      1          num_protect_code_sets : integer;
504:D      1          label_included_flag   : long_boolean;
505:D      1          start_name_sets       : integer;
506:D      1      end;
507:D      1
508:D      1      return_createfile = record
509:D      1          return_mess_header : return_header_type
510:D      1      end;
511:D      1
512:D      1
513:S      1      (+-----+
514:S      1      | Send and return record layouts for request: Create A Link (Rename) |
515:D      1      +-----+)
516:D      1
517:D      1      send_createlink = record
518:D      1          send_mess_header      : send_header_type;
519:D      1          volume_name_header    : volume_header_type;
520:D      1          old_file_name_header  : file_header_type;
521:D      1          new_file_name_header  : file_header_type;
522:D      1          purge_old_link       : long_boolean;
523:D      1          start_name_sets       : integer;
524:D      1      end;
525:D      1
526:D      1      return_createlink = record
527:D      1          return_mess_header : return_header_type;
528:D      1      end;
529:D      1
530:D      1
531:S      1      (+-----+
532:S      1      | Send and return record layouts for request: Exchange files |
533:D      1      +-----+)
534:D      1
535:D      1      send_xchg_data = record
536:D      1          send_mess_header      : send_header_type;
537:D      1          volume_name_header    : volume_header_type;
538:D      1          file_name_header1     : file_header_type;

```

```

539:D      1          file_name_header2    : file_header_type;
540:D      1          start_name_sets         : integer;
541:D      1      end;
542:D      1
543:D      1      return_xchg_data = record
544:D      1          return_mess_header : return_header_type;
545:D      1      end;
546:D      1
547:D      1
548:D      1
549:S      1      (+-----+
550:S      1      | Send and return record layouts for request: Exchange open files |
551:D      1      +-----+)
552:D      1
553:D      1      send_xchg_open = record
554:D      1          send_mess_header      : send_header_type;
555:D      1          file_id_1             : file_id_type;
556:D      1          file_id_2             : file_id_type;
557:D      1      end;
558:D      1
559:D      1      return_xchg_open = record
560:D      1          return_mess_header : return_header_type;
561:D      1      end;
562:D      1
563:D      1
564:D      1
565:S      1      (+-----+
566:S      1      | Send and return record layouts for request: File Information |
567:D      1      +-----+)
568:D      1
569:D      1      send_fileinfo = record
570:D      1          send_mess_header      : send_header_type;
571:D      1          implicit_unlock       : long_boolean;
572:D      1          file_id               : file_id_type;
573:D      1      end;
574:D      1
575:D      1      return_fileinfo = record
576:D      1          return_mess_header : return_header_type;
577:D      1          current_record       : integer;
578:D      1          file_info            : file_info_type;
579:D      1      end;
580:D      1
581:D      1
582:D      1
583:S      1      (+-----+
584:S      1      | Send and return record layouts for request: Ganglia Cleanup |
585:D      1      +-----+)
586:D      1
587:D      1      send_gang_cleanup = record
588:D      1          send_mess_header      : send_header_type;
589:D      1          keep_protected_directories : long_boolean;
590:D      1      end;
591:D      1
592:D      1      return_gang_cleanup = record
593:D      1          return_mess_header : return_header_type
594:D      1      end;
595:D      1
596:D      1
597:D      1
598:S      1      (+-----+

```



```
599:S      | Send and return record layouts for request: Initialize A Volume      |
600:D      |-----|
601:D      |
602:D      | send_initialize = record
603:D      |   send_mess_header      : send_header_type;
604:D      |   volume_name_header    : volume_header_type;
605:D      |   password              : name_type;
606:D      |   num_protect_code_sets : integer;
607:D      |   new_volume_name       : name_type;
608:D      |   new_cat_organization   : name_type;
609:D      |   sectors_per_block     : integer;
610:D      |   format_media          : long_boolean;
611:D      |   interleave_factor     : integer;
612:D      |   filler3               : gang_32bit_filler;
613:D      |   physical_sector_size  : integer;
614:D      |   new_password          : (global_)name_type;
615:D      |   start_protect_code_sets : integer;
616:D      |   init_root_password    : (global_)name_type;
617:D      | end;
618:D      |
619:D      | return_initialize = record
620:D      |   return_mess_header : return_header_type
621:D      | end;
622:D      |
623:D      |
624:S      |
625:S      |
626:D      |
627:D      |
628:D      | send_lockafile = record
629:D      |   send_mess_header : send_header_type;
630:D      |   file_id          : file_id_type;
631:D      |   wait_for_lock    : long_boolean
632:D      | end;
633:D      |
634:D      | return_lockafile = record
635:D      |   return_mess_header : return_header_type;
636:D      |   success            : long_boolean
637:D      | end;
638:D      |
639:D      |
640:S      |
641:S      |
642:D      |
643:D      |
644:D      | send_openafile = record
645:D      |   send_mess_header      : send_header_type;
646:D      |   volume_name_header    : volume_header_type;
647:D      |   file_name_header      : file_header_type;
648:D      |   filler2               : integer;
649:D      |   filler3               : integer;
650:D      |   share_code            : integer;
651:D      |   filler4               : owner_id_type;
652:D      |   filler1               : gang_16bit_filler;
653:D      |   open_type             : gang_open_type;
654:D      |   start_name_sets       : integer
655:D      | end;
656:D      |
657:D      | return_openafile = record
658:D      |   return_mess_header : return_header_type;
```

```
659:D      |   file_id              : file_id_type;
660:D      |   record_mode          : integer;
661:D      |   max_record_size      : integer;
662:D      |   max_file_size        : integer;
663:D      |   file_code            : gang_file_codes;
664:D      |   open_logical_eof     : integer;
665:D      |   sharebits            : integer;
666:D      |   sec_ext_size         : integer;
667:D      |   boot_start_address  : integer;
668:D      | end;
669:D      |
670:D      |
671:S      |
672:S      |
673:D      |
674:D      |
675:D      | send_positiontoarecord = record
676:D      |   send_mess_header : send_header_type;
677:D      |   implicit_unlock  : long_boolean;
678:D      |   file_id          : file_id_type;
679:D      |   filler3          : gang_16bit_filler;
680:D      |   type_of_position : position_type;
681:D      |   byte_offset      : integer
682:D      | end;
683:D      |
684:D      | return_positiontoarecord = record
685:D      |   return_mess_header : return_header_type
686:D      | end;
687:D      |
688:D      |
689:S      |
690:S      |
691:D      |
692:D      |
693:D      | send_purgelink = record
694:D      |   send_mess_header : send_header_type;
695:D      |   volume_name_header : volume_header_type;
696:D      |   file_name_header  : file_header_type;
697:D      |   start_name_sets   : integer
698:D      | end;
699:D      |
700:D      | return_purgelink = record
701:D      |   return_mess_header : return_header_type
702:D      | end;
703:D      |
704:D      |
705:S      |
706:S      |
707:D      |
708:D      |
709:D      | send_readarecord = record
710:D      |   send_mess_header : send_header_type;
711:D      |   implicit_unlock  : long_boolean;
712:D      |   file_id          : file_id_type;
713:D      |   access_code      : integer;
714:D      |   filler3          : array[1..2] of gang_32bit_filler;
715:D      |   requested        : integer;
716:D      |   offset           : integer
717:D      | end;
718:D      |
```



```

719:D 1 return_readarecord = record
720:D 1   return_mess_header : return_header_type;
721:D 1   actual              : integer;
722:D 1   filler1            : array [1..4] of gang_32bit_filler;
723:D 1   data               : packed array[1..512] of char;
724:D 1 end;
725:D 1
726:S 1
727:S 1
728:S 1 (+-----+
729:D 1 | Send and return record layouts for request: Set Date |
730:D 1 +-----+)
731:D 1
732:D 1 send_set_date = record
733:D 1   send_mess_header : send_header_type;
734:D 1   implicit_unlock  : long_boolean;
735:D 1   file_id          : file_id_type;
736:D 1   set_access_date  : long_boolean;
737:D 1   access_date      : date_type;
738:D 1   set_creation_date: long_boolean;
739:D 1   creation_date    : date_type;
740:D 1 end;
741:D 1
742:D 1 return_set_date = record
743:D 1   return_mess_header : return_header_type;
744:D 1 end;
745:D 1
746:S 1 (+-----+
747:D 1 | Send and return record layouts for request: Set Eof |
748:D 1 +-----+)
749:D 1
750:D 1 send_set_eof = record
751:D 1   send_mess_header : send_header_type;
752:D 1   implicit_unlock  : long_boolean;
753:D 1   file_id          : file_id_type;
754:D 1   use_current_ptr  : long_boolean;
755:D 1   byte_offset      : integer;
756:D 1 end;
757:D 1
758:D 1 return_set_eof = record
759:D 1   return_mess_header : return_header_type;
760:D 1 end;
761:D 1
762:D 1
763:S 1 (+-----+
764:D 1 | Send and return record layouts for request: Unlock A File |
765:D 1 +-----+)
766:D 1
767:D 1 send_unlockafile = record
768:D 1   send_mess_header : send_header_type;
769:D 1   file_id          : file_id_type;
770:D 1   explicit_unlock  : long_boolean;
771:D 1 end;
772:D 1
773:D 1 return_unlockafile = record
774:D 1   return_mess_header : return_header_type;
775:D 1 end;
776:D 1
777:D 1
778:S 1 (+-----+

```

```

779:S 1 | Send and return record layouts for request: Volume Status |
780:D 1 +-----+)
781:D 1
782:D 1 send_volume_status = record
783:D 1   send_mess_header : send_header_type;
784:D 1   volume_name_header : volume_header_type;
785:D 1 end;
786:D 1
787:D 1 return_volume_status = record
788:D 1   return_mess_header : return_header_type;
789:D 1   exist              : long_boolean;
790:D 1   interleave         : integer;
791:D 1   volume_name        : name_type;
792:D 1 end;
793:D 1
794:S 1 (+-----+
795:D 1 | Send and return record layouts for request: Write A Record |
796:D 1 +-----+)
797:D 1
798:D 1 send_writearecord = record
799:D 1   send_mess_header : send_header_type;
800:D 1   implicit_unlock  : long_boolean;
801:D 1   file_id          : file_id_type;
802:D 1   access_code      : integer;
803:D 1   filler3          : array [1..2] of gang_32bit_filler;
804:D 1   requested        : integer;
805:D 1   offset           : integer;
806:D 1   filler8          : long_boolean;
807:D 1   flush_buffer     : long_boolean;
808:D 1   data             : packed array [1..512] of char;
809:D 1 end;
810:D 1
811:D 1 return_writearecord = record
812:D 1   return_mess_header : return_header_type;
813:D 1   actual              : integer;
814:D 1 end;
815:D 1
816:D 1
817:D 1
818:D 1
819:S 1 {~~~~~}
820:D 1 $include 'PACKETS'$
821:S 1
822:D 1 type
823:D 1   ptrmp = msg_packet_ptr;
824:D 1   ptrsareyoualive = ^send_are_you_alive;
825:D 1   ptrrareyoualive = ^return_are_you_alive;
826:D 1   ptrscat = ^send_catalogue;
827:D 1   ptrrcat = ^return_catalogue;
828:D 1   ptrscatpass = ^send_cat_passwords;
829:D 1   ptrrcatpass = ^return_cat_passwords;
830:D 1   ptrschangeprotect = ^send_change_protect_codes;
831:D 1   ptrrchangeprotect = ^return_change_protect_codes;
832:D 1   ptrschangevolume = ^send_change_volume_label;
833:D 1   ptrrchangevolume = ^return_change_volume_label;
834:D 1   ptrsclosefile = ^send_closefile;
835:D 1   ptrrclose = ^return_closefile;
836:D 1   ptrscopy = ^send_copy;
837:D 1   ptrrcopy = ^return_copy;
838:D 1   ptrscreatefile = ^send_createfile;

```



```

839:D 1 ptrrcreatefile = ^return_createfile;
840:D 1 ptrrcreatelink = ^send_createfile;
841:D 1 ptrrcreatelink = ^return_createfile;
842:D 1 ptrrexchange = ^send_xchg_open;
843:D 1 ptrrexchange = ^return_xchg_open;
844:D 1 ptrrfileinfo = ^send_fileinfo;
845:D 1 ptrrfileinfo = ^return_fileinfo;
846:D 1 ptrrgangclean = ^send_gang_cleanup;
847:D 1 ptrrgangclean = ^return_gang_cleanup;
848:D 1 ptrrsinit = ^send_initialize;
849:D 1 ptrrsinit = ^return_initialize;
850:D 1 ptrrlock = ^send_lockfile;
851:D 1 ptrrlock = ^return_lockfile;
852:D 1 ptrsopen = ^send_opensfile;
853:D 1 ptrsopen = ^return_opensfile;
854:D 1 ptrspos = ^send_positiontoarecord;
855:D 1 ptrspos = ^return_positiontoarecord;
856:D 1 ptrspurge = ^send_purgefile;
857:D 1 ptrspurge = ^return_purgefile;
858:D 1 ptrsread = ^send_readarecord;
859:D 1 ptrsread = ^return_readarecord;
860:D 1 ptrssetdate = ^send_set_date;
861:D 1 ptrssetdate = ^return_set_date;
862:D 1 ptrsseteof = ^send_set_eof;
863:D 1 ptrsseteof = ^return_set_eof;
864:D 1 ptrsunlock = ^send_unlockfile;
865:D 1 ptrsunlock = ^return_unlockfile;
866:D 1 ptrsvol = ^send_volume_status;
867:D 1 ptrsvol = ^return_volume_status;
868:D 1 ptrswrite = ^send_writearecord;
869:D 1 ptrswrite = ^return_writearecord;
870:S
871:D 1 ptrshead = ^send_header_type;
872:D 1 ptrshead = ^return_header_type;
873:S
874:D 1 pk_ptr = record case integer of
875:D 1 0 : (mp : ptrmp);
876:D 1 1 : (sareyoualive : ptrsareyoualive);
877:D 1 2 : (rareyoualive : ptrrareyoualive);
878:D 1 3 : (scat : ptrscat);
879:D 1 4 : (rcat : ptrrcat);
880:D 1 5 : (scatpass : ptrscatpass);
881:D 1 6 : (rcatpass : ptrrcatpass);
882:D 1 7 : (schangeprotect : ptrschangeprotect);
883:D 1 8 : (rchangeprotect : ptrrchangeprotect);
884:D 1 9 : (schangevolume : ptrschangevolume);
885:D 1 10 : (rchangevolume : ptrrchangevolume);
886:D 1 11 : (sclose : ptrsclose);
887:D 1 12 : (rclose : ptrrclose);
888:D 1 13 : (scopy : ptrscopy);
889:D 1 14 : (rcopy : ptrrcopy);
890:D 1 15 : (screatefile : ptrscreatefile);
891:D 1 16 : (rcreatefile : ptrrcreatefile);
892:D 1 17 : (screatelink : ptrscreatelink);
893:D 1 18 : (rcreatelink : ptrrcreatelink);
894:D 1 19 : (sexchange : ptrsexchange);
895:D 1 20 : (rexchange : ptrrexchange);
896:D 1 21 : (sfileinfo : ptrsfileinfo);
897:D 1 22 : (rfileinfo : ptrrfileinfo);
898:D 1 23 : (sgangclean : ptrsgangclean);

```

```

899:D 1 24 : (rgangclean : ptrrgangclean);
900:D 1 25 : (sinit : ptrsinit);
901:D 1 26 : (rinit : ptrrinit);
902:D 1 27 : (slock : ptrslock);
903:D 1 28 : (rlock : ptrrlock);
904:D 1 29 : (sopen : ptrsopen);
905:D 1 30 : (ropen : ptrropen);
906:D 1 31 : (spos : ptrspos);
907:D 1 32 : (rpos : ptrrpos);
908:D 1 33 : (spurge : ptrspurge);
909:D 1 34 : (rpurge : ptrrpurge);
910:D 1 35 : (sread : ptrsread);
911:D 1 36 : (rread : ptrrread);
912:D 1 37 : (ssetdate : ptrssetdate);
913:D 1 38 : (rsetdate : ptrrsetdate);
914:D 1 39 : (sseteof : ptrsseteof);
915:D 1 40 : (rseteof : ptrrseteof);
916:D 1 41 : (sunlock : ptrsunlock);
917:D 1 42 : (runlock : ptrrunlock);
918:D 1 43 : (svol : ptrsvol);
919:D 1 44 : (rvol : ptrrvol);
920:D 1 45 : (swrite : ptrswrite);
921:D 1 46 : (rwrite : ptrrwrite);
922:D 1 47 : (shead : ptrshead);
923:D 1 48 : (rhead : ptrrhead);
924:D 1 end;
925:S
926:D 1 name_set_array_three = array [1..3] of file_name_set;
927:D 1 name_set_array = array [1..6] of file_name_set;
928:D 1 pnsa = ^name_set_array;
929:S
930:D 1 protectcode_set_array = array [1..24] of protect_code_set;
931:D 1 ppsa = ^protectcode_set_array;
932:S
933:D 1 $include 'INIT:SRM_TYPES'S
934:D 1 $include 'INIT:SRM_ERRS'S
935:D 1 CONST
936:D 1 ios_error_base = 31000;
937:D 1 ios_software_bug = 31000;
938:D 1 ios_bad_select_code = 31001;
939:D 1 ios_unallocated_extent = 31002;
940:D 1 ios_ds_rom_missing = 31003;
941:D 1 ios_unsupported_dam = 31004;
942:D 1 ios_device_drivers_dont_match = 31005;
943:D 1 ios_invalid_ios_request = 31006;
944:D 1 ios_attach_table_full = 31007;
945:D 1 ios_improper_mass_storage_device = 31008;
946:D 1 ios_directory_formats_dont_match = 31009;
947:D 1 ios_invalid_file_size = 31010;
948:D 1 ios_invalid_file_id = 31011;
949:D 1 ios_volume_recoverable_error = 31012;
950:D 1 ios_volume_io_error = 31013;
951:D 1 ios_file_pathname_missing = 31014;
952:D 1 ios_illegal_byte_number = 31015;
953:D 1 ios_corrupt_directory = 31016;
954:D 1 ios_successful_completion = 31017;
955:D 1 ios_system_down = 31018;
956:D 1 ios_file_unopened = 31019;
957:D 1 ios_volume_offline = 31020;
958:D 1 ios_volume_labels_dont_match = 31021;

```



```

958:D 1 ios_password_not_allowed = 31022;
959:D 1 ios_access_to_file_not_allowed = 31023;
960:D 1 ios_unsupported_directory_operation = 31024;
961:D 1 ios_conflicting_share_modes = 31025;
962:D 1 ios_bad_file_name = 31026;
963:D 1 ios_file_in_use = 31027;
964:D 1 ios_insufficient_disk_space = 31028;
965:D 1 ios_duplicate_filenames = 31029;
966:D 1 ios_phys_eof_encountered = 31030;
967:D 1 ios_no_capability_for_file = 31031;
968:D 1 ios_file_not_found = 31032;
969:D 1 ios_volume_in_use = 31033;
970:D 1 ios_file_not_directory = 31034;
971:D 1 ios_directory_not_empty = 31035;
972:D 1 ios_volume_not_found = 31036;
973:D 1 ios_invalid_protect_code = 31037;
974:D 1 ios_volume_unrecoverable_error = 31038;
975:D 1 ios_password_not_found = 31039;
976:D 1 ios_duplicate_passwords = 31040;
977:D 1 ios_deadlock_detected = 31041;
978:D 1 ios_link_to_directory_not_allowed = 31042;
979:D 1 ios_rename_across_volumes = 31043;
980:D 1 ios_volume_down = 31044;
981:D 1 ios_eof_encountered = 31045;
982:D 1 ios_invalid_file_code = 31046;
983:D 1 ios_file_locked_please_retry = 31047;
984:D 1 ios_no_reply = 31048;
985:D 1 ios_purge_on_open = 31049;
986:S
987:D 1 ios_error_top = 31049;
988:D 1 ( -----' errors start at 31050 )
989:S
990:D 1 $include 'INIT:SRM_ERRS'$
991:S
992:D 1 var
993:D -4 1 packet_ptr : pk_ptr;
994:D -8 1 defaulttimeout : integer; (timeout values in milliseconds)
995:D -12 1 waitforlocktimeout : integer;
996:D -16 1 copytimeout : integer;
997:D -18 1 srmsavesc : shortint;
998:S
999:D -18 1 procedure srm_init;
1000:D -18 1 procedure resetcard(unum : unitnum);
1001:D -18 1 procedure packetout(unum : unitnum);
1002:D -18 1 procedure packetin(unum : unitnum);
1003:D -18 2 sendreq : integer;
1004:D -18 1 procedure areyoualivepack(unum : unitnum);
1005:D -18 1 procedure catpack(unum : unitnum);
1006:D 2 nfns : integer;
1007:D 2 nsaptr : pnsa;
1008:D 2 path : path_start_type;
1009:D 2 wd : file_id_type;
1010:D 2 rtpass : name_type;
1011:D 2 max : integer;
1012:D -18 2 indx : integer;
1013:D -18 1 procedure catpasspack(unum : unitnum);
1014:D 2 nfns : integer;
1015:D 2 nsaptr : pnsa;
1016:D 2 path : path_start_type;
1017:D 2 wd : file_id_type;

```

```

1018:D 2 rtpass : name_type;
1019:D 2 max : integer;
1020:D -18 2 indx : integer;
1021:D -18 1 procedure changeprotectpack(unum : unitnum);
1022:D 2 nfns : integer;
1023:D 2 nsaptr : pnsa;
1024:D 2 path : path_start_type;
1025:D 2 wd : file_id_type;
1026:D 2 rtpass : name_type;
1027:D 2 nps : integer;
1028:D -18 2 psaptr : ppsa;
1029:D -18 1 procedure changevolpack(unum : unitnum);
1030:D 2 vpass : name_type;
1031:D 2 newname : name_type;
1032:D -18 2 newpass : name_type;
1033:D -18 1 procedure closepack(unum : unitnum);
1034:D -18 2 fid : file_id_type;
1035:D -18 1 procedure copenpack(unum : unitnum);
1036:D 2 srcfid : file_id_type;
1037:D 2 srcoff : integer;
1038:D 2 destfid : file_id_type;
1039:D 2 destoff : integer;
1040:D -18 2 req : integer;
1041:D -18 1 procedure createpack(unum : unitnum);
1042:D 2 nfns : integer;
1043:D 2 nsaptr : pnsa;
1044:D 2 path : path_start_type;
1045:D 2 wd : file_id_type;
1046:D 2 rtpass : name_type;
1047:D 2 nps : integer;
1048:D 2 psaptr : ppsa;
1049:D 2 ftype : gang_file_codes;
1050:D 2 mode : integer;
1051:D 2 maxrec : integer;
1052:D 2 ext1 : integer;
1053:D 2 ext2 : integer;
1054:D -18 2 xaddr : integer;
1055:D -18 1 procedure createlinkpack(unum : unitnum);
1056:D 2 oldnfns : integer;
1057:D 2 oldnsaptr : pnsa;
1058:D 2 oldpath : path_start_type;
1059:D 2 oldwd : file_id_type;
1060:D 2 oldrtpass : name_type;
1061:D 2 newnfns : integer;
1062:D 2 newnsaptr : pnsa;
1063:D 2 newpath : path_start_type;
1064:D 2 newwd : file_id_type;
1065:D 2 newrtpass : name_type;
1066:D -18 2 purgeold : boolean;
1067:D -18 1 procedure exchangeunpack(unum : unitnum);
1068:D 2 fid1 : file_id_type;
1069:D 2 fid2 : file_id_type;
1070:D -18 1 procedure fileinfounpack(unum : unitnum);
1071:D -18 2 fid : file_id_type;
1072:D -18 1 procedure gangcleanpack(unum : unitnum);
1073:D -18 2 savewd : boolean;
1074:D -18 1 procedure lockpack(unum : unitnum);
1075:D 2 fid : file_id_type;
1076:D -18 2 wait : boolean;
1077:D -18 1 procedure openpack(unum : unitnum);

```



```

1078:D      2      nfns      : integer;
1079:D      2      nsaptr    : pnsa;
1080:D      2      path      : path_start_type;
1081:D      2      wd         : file_id_type;
1082:D      2      rtpass     : name_type;
1083:D      2      share      : integer;
1084:D      2      opn        : gang_open_type;
1085:D      -18     procedure pospack(unum : unitnum;
1086:D      2      fid         : file_id_type;
1087:D      2      typepos      : position_type;
1088:D      -18     procedure purgepack(unum : unitnum;
1089:D      2      boffset      : integer;
1090:D      2      nfns         : integer;
1091:D      2      nsaptr        : pnsa;
1092:D      2      path        : path_start_type;
1093:D      2      wd          : file_id_type;
1094:D      -18     procedure sendreadpack(unum : unitnum;
1095:D      2      fid          : file_id_type;
1096:D      2      access         : integer;
1097:D      2      reg           : integer;
1098:D      -18     procedure seteofpack(unum : unitnum;
1099:D      2      off          : integer;
1100:D      2      dat          : anyptr);
1101:D      2      procedure setdefaulttimeout(time : integer);
1102:D      2      procedure setcopytimeout(time : integer);
1103:D      -18     procedure setwaitforlocktimeout(time : integer);
1104:D      2      procedure unlockpack(unum : unitnum;
1105:D      2      fid          : file_id_type);
1106:D      -18     procedure volpack(unum : unitnum;
1107:D      2      fid          : file_id_type;
1108:D      2      access         : integer;
1109:D      2      reg           : integer;
1110:D      2      off          : integer;
1111:D      -18     procedure setdefaulttimeout(time : integer);
1112:D      -18     procedure setcopytimeout(time : integer);
1113:D      -18     procedure setwaitforlocktimeout(time : integer);
1114:D      -18     1 implement
1115:D      -18     1 const
1116:D      -18     1 packetouttimeout = 1000; (timeout values in milliseconds)
1117:D      -18     1 ayatimeout = 1000;
1118:D      -18     1 var
1119:D      -22     1 dumbuf : ^buf_info_type;
1120:D      -23     1 srm_initd : boolean;
1121:D      -24     1 waitingforlock : boolean;
1122:D      -24     1 (*****
1123:D      1 procedure maptoioresult(status : integer),
1124:D      2 begin
1125:D      2 if ioresult = ord(inoerror) then
1126:D      3 if status <> 0 then
1127:D      4 case status of
1128:D      5   ios_bad_select_code : ioresult := ord(ibadunit);
1129:D      5   ios_attach_table_full : ioresult := ord(itoomanyopen);

```

```

1138:C      5   ios_invalid_file_size : ioresult := ord(inotvalidsize);
1139:C      5   ios_invalid_file_id : ioresult := ord(ilstofile);
1140:S
1141:C      5   ios_bad_file_name : ioresult := ord(ibadtitle);
1142:C      5   ios_file_pathname_missing : ioresult := ord(ibadvalue);
1143:S
1144:C      5   ios_illegal_byte_number : ioresult := ord(ibadvalue);
1145:S
1146:C      5   ios_successful_completion,
1147:C      5   ios_no_reply : ioresult := ord(inoerror);
1148:S
1149:C      5   ios_system_down,
1150:C      5   ios_volume_offline,
1151:C      5   ios_volume_not_found,
1152:C      5   ios_volume_down : ioresult := ord(znodevice);
1153:S
1154:C      5   ios_file_unopened : ioresult := ord(inotopen);
1155:S
1156:C      5   ios_password_not_allowed,
1157:C      5   ios_no_capability_for_code,
1158:C      5   ios_invalid_protect_code,
1159:C      5   ios_password_not_found,
1160:C      5   ios_duplicate_passwords : ioresult := ord(ibadpass);
1161:S
1162:C      5   ios_access_to_file_not_allowed : ioresult := ord(inoaccess);
1163:S
1164:C      5   ios_unsupported_directory_operation,
1165:C      5   ios_link_to_directory_not_allowed : ioresult := ord(inotondir);
1166:S
1167:C      5   ios_deadlock_detected,
1168:C      5   ios_conflicting_share_modes,
1169:C      5   ios_file_locked_please_retry : ioresult := ord(ifilelocked);
1170:S
1171:C      5   ios_file_in_use,
1172:C      5   ios_purge_on_open : ioresult := ord(inotclosed);
1173:S
1174:C      5   ios_insufficient_disk_space : ioresult := ord(inoroom);
1175:C      5   ios_duplicate_filenames : ioresult := ord(idupfile);
1176:S
1177:C      5   ios_phys_eof_encountered,
1178:C      5   ios_eof_encountered : ioresult := ord(ieof);
1179:S
1180:C      5   ios_file_not_found : ioresult := ord(inofile);
1181:C      5   ios_volume_in_use : ioresult := ord(znotready);
1182:C      5   ios_file_not_directory : ioresult := ord(ifilenotdir);
1183:C      5   ios_directory_not_empty : ioresult := ord(idirnotempty);
1184:C      5   ios_invalid_file_code : ioresult := ord(ibadfiletype);
1185:S
1186:C      5   otherwise : ioresult := ord(isrmcatchall);
1187:C      2 end;
1188:C      -24 1 (*****
1189:C      1 procedure initdumbuf;
1190:C      2 begin
1191:C      2 with dumbuf^ do
1192:C      3 begin
1193:C      3   buf_ptr := nil;
1194:C      3   act_tfr := no_tfr;
1195:C      3   active_isc := no_isc;

```



```

1198:C      3      buf_size      := 0;
1199:C      3      buf_empty     := nil;
1200:C      3      buf_fill      := nil;
1201:C      3      drv_tmp_ptr    := nil;
1202:C      3      eot_proc.dummy_sl := nil;
1203:C      3      eot_proc.dummy_pr := nil;
1204:C      3      eot_parm       := nil;
1205:C      3      dma_priority    := false;
1206:C      3      end;
1207:C      2 end
1208:D -24 1 (*****
1209:S
1210:D -24 1 (*****
1211:S
1212:D      1 procedure srm_init;
1213:C      2 begin
1214:C      3   if not srm_initd then
1215:C      4     begin
1216:C      5       new(packet_ptr.mp);
1217:C      5       new(dumbuf);
1218:C      5       markuser;
1219:C      5       srm_initd := true;
1220:C      5     end;
1221:C      5   initdumbuf;
1222:C      5   defaulttimeout := 30000;      (timeout values in milliseconds)
1223:C      5   waitforlocktimeout := 0;
1224:C      5   pytimeout := 120000;
1225:C      2 end;
1226:D -24 1 (*****
1227:S
1228:D      1 procedure resetcard(unum : unithum);
1229:C      2 begin
1230:C      3   with isc_table[unithum] do
1231:C      4     call(io_drv_ptr^.iod_init, io_tmp_ptr);
1232:C      3   end;
1233:D -24 1 (*****
1234:S
1235:D      1 procedure setdefaulttimeout(time: integer);
1236:C      2 begin
1237:C      3   defaulttimeout := time;      (time is in milliseconds)
1238:C      2 end;
1239:D -24 1 (*****
1240:S
1241:D      1 procedure setwaitforlocktimeout(time: integer);
1242:C      2 begin
1243:C      3   waitforlocktimeout := time;  (time is in milliseconds)
1244:C      2 end;
1245:D -24 1 (*****
1246:S
1247:D      1 procedure setcopytimeout(time: integer);
1248:C      2 begin
1249:C      3   copytimeout := time;      (time is in milliseconds)
1250:C      2 end;
1251:D -24 1 (*****
1252:S
1253:D      1 procedure setintegertimeout(sc : type_isc;
1254:C      2      time: integer);
1255:C      3 begin
1256:C      4   with isc_table[sc] do
1257:C      5     begin

```

```

1258:C      3   user_time := time;      (time is in milliseconds)
1259:C      3   if io_tmp_ptr <> nil then
1260:C      4     io_tmp_ptr^.timeout := time;
1261:C      3   end;
1262:C      2 end;
1263:D -24 1 (*****
1264:S
1265:D      1 function do_buffer_data(var b_info : buf_info_type) : integer;
1266:C      2 begin
1267:C      3   with b_info do
1268:C      4     do_buffer_data := integer(buf_fill) - integer(buf_empty);
1269:C      3   end;
1270:D -24 1 (*****
1271:S
1272:D      1 procedure do_buffer_reset(var b_info : buf_info_type);
1273:C      2 begin
1274:C      3   with b_info do
1275:C      4     if active_isc <> no_isc then
1276:C      5       io_escape(ioe_buf_busy, no_isc)
1277:C      4     else
1278:C      5       begin
1279:C      6         buf_fill := buf_ptr;
1280:C      6         buf_empty := buf_ptr;
1281:C      5       end;
1282:C      3   end;
1283:D -24 1 (*****
1284:S
1285:D      1 function do_buffer_space(var b_info : buf_info_type) : integer;
1286:C      2 begin
1287:C      3   with b_info do
1288:C      4     begin
1289:C      5       if (do_buffer_data(b_info) = 0)
1290:C      6         and (active_isc = no_isc) then
1291:C      7         do_buffer_reset(b_info);
1292:C      5       do_buffer_space := buf_size + integer(buf_ptr) - integer(buf_fill);
1293:C      4     end;
1294:C      3   end;
1295:D -24 1 (*****
1296:S
1297:D      1 procedure do_transfer(sc : type_isc;
1298:C      2      dir : dir_of_tfr;
1299:C      2      count : integer;
1300:C      2      tfr_end : boolean);
1301:D      3 var
1302:D      4   lcount : integer;
1303:C      2 begin
1304:C      3   with isc_table[sc], dumbuf^ do
1305:C      4     begin
1306:C      5       if not tfr_end then
1307:C      6         lcount := count
1308:C      5       else
1309:C      6         if dir = from_memory then
1310:C      7           lcount := do_buffer_data(dumbuf^);
1311:C      6         else
1312:C      7           lcount := do_buffer_space(dumbuf^);
1313:C      5       end;
1314:C      3   if io_tmp_ptr = nil then
1315:C      4     io_escape(ioe_no_driver, sc);
1316:C      3   if lcount = 0 then
1317:C      4     io_escape(ioe_bad_cnt, no_isc);

```



```
1318:C      3      if active_isc <> no_isc then
1319:C      4          io_escape(ioe_buf_busy,no_isc);
1320:S
1321:C      3      if do_buffer_data(dumbuf) = 0 then
1322:C      4          do_buffer_reset(dumbuf);
1323:S
1324:C      3      with io_tmp_ptr^ do
1325:C      4          if dir = to_memory then
1326:C      5              begin
1327:C      6                  if in_bufptr <> nil then
1328:C      7                      io_escape(ioe_isc_busy,sc);
1329:C      8                      if do_buffer_space(dumbuf) < lcount then
1330:C      9                          io_escape(ioe_no_space,sc);
1331:C      10                         in_bufptr := dumbuf;
1332:C      11                     end
1333:C      12                     else
1334:C      13                         begin
1335:C      14                             if out_bufptr <> nil then
1336:C      15                                 io_escape(ioe_isc_busy,sc);
1337:C      16                                 if do_buffer_data(dumbuf) < lcount then
1338:C      17                                     io_escape(ioe_no_data,sc);
1339:C      18                                     out_bufptr := dumbuf;
1340:C      19                                 end;
1341:S
1342:C      3          drv_tmp_ptr := io_tmp_ptr;
1343:C      3          act_tfr := no_tfr;
1344:C      3          usr_tfr := seflast_fastest;
1345:C      3          b_w_mode := false; (byte mode)
1346:C      3          direction := dir;
1347:C      3          term_char := -1; (no term char)
1348:C      3          term_count := lcount;
1349:C      3          end_mode := tfr_end;
1350:S
1351:C      3          call(io_drv_ptr^.iod_tfr,io_tmp_ptr,dumbuf);
1352:C      3          end;
1353:C      2      end;
1354:D      -24      1      (*****
1355:S
1356:D      1      procedure dorecover(unum : unitnum);
1357:C      2      begin
1358:C      3          ioresult := ord(isrmcatchall);
1359:C      4          if (escapecode <> ioescapecode) then
1360:C      5              begin
1361:C      6                  if srmsavesc = 0 then
1362:C      7                      srmsavesc := escapecode
1363:C      8                  end
1364:C      9                  else
1365:C      10                     if (ioe_result = ioe_timeout) then
1366:C      11                         ioresult := ord(ztimeout);
1367:C      12                     resetcard(unum);
1368:C      13                 end;
1369:D      -24      1      (*****
1370:S
1371:D      1      procedure packetout(unum : unitnum);
1372:D      2      var
1373:D      3          ip : ^integer;
1374:C      -4      2      begin
1375:C      3          with unitable^[unum] do
1376:C      4              try
1377:C      5                  initdumbuf;
```

```
1378:C      4      setintegertimeout(sc,packetouttimeout);
1379:C      4      packet_ptr.mp^[8] := chr(unitable^[unum].ba);
1380:C      4      packet_ptr.mp^[9] := chr(0);
1381:C      4      packet_ptr.mp^[10] := chr(0);
1382:C      4      packet_ptr.mp^[11] := chr(0); (request type)
1383:C      4      ip := addr(packet_ptr.mp^[13]); (packet length)
1384:C      5      if ip^ = req_are_you_alive then
1385:C      6          packet_ptr.mp^[12] := chr(2)
1386:C      7      else
1387:C      8          packet_ptr.mp^[12] := chr(7);
1388:C      9      dumbuf^.buf_ptr := anyptr(packet_ptr.mp);
1389:C      4      dumbuf^.buf_size := sizeof(msg_packet_type);
1390:C      4      dumbuf^.buf_empty := addr(packet_ptr.mp^[8]);
1391:C      4      ip := addr(packet_ptr.mp^[13]);
1392:C      4      dumbuf^.buf_fill := addr(packet_ptr.mp^[12+1+ip]);
1393:C      4      do_transfer(sc,from_memory,0,true);
1394:C      4      recover
1395:C      4      dorecover(unum);
1396:C      2      end;
1397:D      -24      1      (*****
1398:S
1399:D      1      procedure packetin(unum : unitnum;
1400:D      2                          sendreq : integer);
1401:D      2      type
1402:D      3          ayastatus_type = packed record
1403:D      4              srmnode : byte;
1404:D      5              linkerrs : byte;
1405:D      6              computerid : shortint;
1406:D      7          end;
1407:D      2      ayastatus_ptr = ^ayastatus_type;
1408:D      2      var
1409:D      3          count : integer;
1410:C      -4      2      begin
1411:C      3          with unitable^[unum] do
1412:C      4              try
1413:C      5                  initdumbuf;
1414:C      6                  if sendreq = req_are_you_alive then
1415:C      7                      setintegertimeout(sc,ayatimeout)
1416:C      8                  else if (sendreq = req_flock) and (waitingforlock) then
1417:C      9                      setintegertimeout(sc,waitforlocktimeout)
1418:C      10                     else if (sendreq = req_copy) or
1419:C      11                         (sendreq = req_create) then { 3.0 BUG FIX 3/16/84 }
1420:C      12                         setintegertimeout(sc,copytimeout)
1421:C      13                     else
1422:C      14                         setintegertimeout(sc,defaulttimeout);
1423:C      15                     with packet_ptr.rhead^, packet_ptr.rread^ do
1424:C      16                         begin
1425:C      17                             repeat
1426:C      18                                 fillchar(linkfiller, 28, chr(0));
1427:S
1428:C      6                             dumbuf^.buf_ptr := anyptr(packet_ptr.mp);
1429:C      6                             dumbuf^.buf_size := sizeof(msg_packet_type);
1430:C      6                             dumbuf^.buf_empty := anyptr(packet_ptr.mp);
1431:C      6                             dumbuf^.buf_fill := addr(packet_ptr.mp^[9]);
1432:S
1433:C      6                             if sendreq <> req_read then
1434:C      7                                 begin
1435:C      8                                     do_transfer(sc,to_memory,0,true);
1436:C      9                                 end
1437:C      10                             else
```



```
1438:C      7      begin
1439:C      7      count := size_from_gang_error + 4;
1440:C      7      do_transfer(sc,to_memory,count,false);
1441:S      7
1442:C      7      if message_length > size_from_gang_error then
1443:C      8      if return_request_type <> -req_read then
1444:C      9      begin
1445:C      9      do_transfer(sc,to_memory,0,true);
1446:C      9      end
1447:C      9      else
1448:C      9      begin
1449:C      9      count := size_from_read - size_from_gang_error;
1450:C      9      do_transfer(sc,to_memory,count,false);
1451:S      9
1452:C      9      count := actual;
1453:S      9
1454:C      9      if count > 0 then
1455:C      10      begin
1456:C      10      dumbuf^.buf_ptr := anyptr(user_sequencing_field);
1457:C      10      dumbuf^.buf_size := 512;
1458:C      10      dumbuf^.buf_empty := anyptr(user_sequencing_field);
1459:C      10      dumbuf^.buf_fill := anyptr(user_sequencing_field);
1460:C      10      do_transfer(sc,to_memory,count,false);
1461:C      10      end;
1462:C      9      end;
1463:C      7      end;
1464:C      6      until return_request_type = -sendreq;
1465:S      6
1466:C      6      if sendreq <> req_are_you_alive then
1467:C      6      ioretoresult(status);
1468:C      6      else
1469:C      6      if ayastatusptr(addr(status))^.srnode <> 1 then
1470:C      7      ioretoresult := ord(znodevice);
1471:C      5      end;
1472:C      4      recover
1473:C      4      dorecover(unum);
1474:C      2      end;
1475:D      -24 1 (*****
1476:S
1477:D      1      procedure setup_smh(var smh      : send_header_type;
1478:D      2      ml,
1479:D      2      srt,
1480:D      2      usf      : integer);
1481:C      2      begin
1482:C      3      with smh do
1483:C      3      begin
1484:C      3      message_length := ml;
1485:C      3      send_request_type := srt;
1486:C      3      user_sequencing_field := usf;
1487:C      3      end;
1488:C      2      end;
1489:D      -24 1 (*****
1490:S
1491:D      1      procedure setup_vnh(var vnh      : volume_header_type;
1492:D      2      unum      : unitnum);
1493:C      2      begin
1494:C      3      with vnh do
1495:C      3      begin
1496:C      3      filler1 := 0;
1497:C      3      driver_name := ' ';
```

```
1498:C      3      catalogue_organization := ' ';
1499:C      3      device_address_present.i := 1;
1500:C      3      with device_address do
1501:C      4      begin
1502:C      4      address1 := unitable^[unum].du; {unit number}
1503:C      4      haddress := 0;
1504:C      4      unit_num := 0;
1505:C      4      volume_num := 0;
1506:C      4      end;
1507:C      3      volume_name := ' ';
1508:C      3      end;
1509:C      2      end;
1510:D      -24 1 (*****
1511:S
1512:D      1      procedure setup_fnh(var fnh      : file_header_type;
1513:D      2      num      : integer;
1514:D      2      wd      : file_id_type;
1515:D      2      pt      : path_start_type;
1516:D      2      rp      : name_type);
1517:C      2      begin
1518:C      3      with fnh do
1519:C      3      begin
1520:C      3      num_file_name_sets := num;
1521:C      3      working_directory := wd;
1522:C      3      filler1 := 0;
1523:C      3      path_type := pt;
1524:C      3      root_password := rp;
1525:C      3      end;
1526:C      2      end;
1527:D      -24 1 (*****
1528:S
1529:D      1      procedure areyoualivepack(unum      : unitnum);
1530:C      2      begin
1531:C      3      with packet_ptr.sareyoualive^ do
1532:C      3      begin
1533:C      3      setup_smh(send_mess_header,
1534:C      3      size_to_are_you_alive,
1535:C      3      req_are_you_alive,
1536:C      3      0);
1537:C      3      end;
1538:S      3
1539:C      2      packetout(unum);
1540:C      2      if ioretoresult = ord(ioerror) then
1541:C      3      packetin(unum,req_are_you_alive);
1542:C      2      if ioretoresult <> ord(ioerror) then
1543:C      3      packet_ptr.rareyoualive^.return_mess_header.status := 0;
1544:C      2      end;
1545:D      -24 1 (*****
1546:S
1547:D      1      procedure catpack(unum      : unitnum;
1548:D      2      nfns      : integer;
1549:D      2      nsaptr      : pnsa;
1550:D      2      path      : path_start_type;
1551:D      2      wd      : file_id_type;
1552:D      2      rtpass      : name_type;
1553:D      2      max      : integer;
1554:D      2      indx      : integer);
1555:C      -16 2      begin
1556:C      2      with packet_ptr.scata^ do
1557:C      3      begin
```



```

1558:C      3      pnsa(addr(start_name_sets))^ := nsaptr^;
1559:C
1560:C      3      setup_smh(send_mess_header,
1561:C      3      size_to_cat+nfns*36,
1562:C      3      req_catalog,
1563:C      3      0);
1564:C
1565:C      3      max_num_files := max;
1566:C      3      file_index := indx;
1567:C      3      filler1 := 0;
1568:C
1569:C      3      setup_vnh(volume_name_header,unum);
1570:C
1571:C      3      filler2 := 0;
1572:C
1573:C      3      setup_fnh(file_name_header,
1574:C      3      nfns,
1575:C      3      wd, (working directory)
1576:C      3      path,
1577:C      3      rtpass);(root password)
1578:C
1579:C      3      end;
1580:C      2      packetout(unum);
1581:C      2      packetin(unum,req_catalog);
1582:C      2      end;
1583:D      -24 1 (*****
1584:C
1585:D      1      procedure catpasspack(unum      : unitnum;
1586:D      2      nfns      : integer;
1587:D      2      nsaptr    : pnsa;
1588:D      2      path      : path_start_type;
1589:D      2      wd        : file_id_type;
1590:D      2      rtpass    : name_type;
1591:D      2      max       : integer;
1592:D      2      indx     : integer);
1593:D      -16 2      begin
1594:C      2      with packet_ptr.scapass^ do
1595:C      3      begin
1596:C      3      pnsa(addr(start_name_sets))^ := nsaptr^;
1597:C
1598:C      3      setup_smh(send_mess_header,
1599:C      3      size_to_catprotect+nfns*36,
1600:C      3      req_catprotect,
1601:C      3      0);
1602:C
1603:C      3      max_num_passwords := max;
1604:C      3      filler1 := 0;
1605:C      3      password_index := indx;
1606:C
1607:C      3      setup_vnh(volume_name_header,unum);
1608:C
1609:C      3      filler2 := 0;
1610:C
1611:C      3      setup_fnh(file_name_header,
1612:C      3      nfns,
1613:C      3      wd, (working directory)
1614:C      3      path,
1615:C      3      rtpass);(root password)
1616:C
1617:C      3      end;

```

```

1618:C      2      packetout(unum);
1619:C      2      packetin(unum,req_catprotect);
1620:C      2      end;
1621:D      -24 1 (*****
1622:C
1623:D      1      procedure changeprotectpack(unum      : unitnum;
1624:D      2      nfns      : integer;
1625:D      2      nsaptr    : pnsa;
1626:D      2      path      : path_start_type;
1627:D      2      wd        : file_id_type;
1628:D      2      rtpass    : name_type;
1629:D      2      nps      : integer;
1630:D      2      psaptr   : ppsa);
1631:D      -16 2      begin
1632:C      2      with packet_ptr.schangeprotect^ do
1633:C      3      begin
1634:C      3      pnsa(addr(start_name_sets))^ := nsaptr^;
1635:C      3      ppsa(addr(pnsa(addr(start_name_sets))[nfns+1]))^ := psaptr^;
1636:C
1637:C      3      setup_smh(send_mess_header,
1638:C      3      size_to_changeprotect+nfns*36+nps*24,
1639:C      3      req_changeprotect,
1640:C      3      0);
1641:C
1642:C      3      setup_vnh(volume_name_header,unum);
1643:C
1644:C      3      setup_fnh(file_name_header,
1645:C      3      nfns,
1646:C      3      wd, (working directory)
1647:C      3      path,
1648:C      3      rtpass);(root password)
1649:C
1650:C      3      num_protect_code_sets := nps;
1651:C      3      end;
1652:C
1653:C      2      packetout(unum);
1654:C      2      packetin(unum,req_changeprotect);
1655:C      2      end;
1656:D      -24 1 (*****
1657:C
1658:D      1      procedure changevolpack(unum      : unitnum;
1659:D      2      vpass     : name_type;
1660:D      2      newname    : name_type;
1661:D      2      newpass    : name_type);
1662:D      -48 2      begin
1663:C      2      with packet_ptr.schangevolume^ do
1664:C      3      begin
1665:C      3      setup_smh(send_mess_header,
1666:C      3      size_to_change_vol_label,
1667:C      3      req_label,
1668:C      3      0);
1669:C
1670:C      3      setup_vnh(volume_name_header,unum);
1671:C      3      password := vpass;
1672:C      3      new_volume_name := newname;
1673:C      3      new_vol_password := newpass;
1674:C      3      end;
1675:C
1676:C      2      packetout(unum);
1677:C      2      packetin(unum,req_label);

```



```
1678:C      2 end;
1679:D -24 1 (*****
1680:S
1681:D      1 procedure closepack(unum      : unitnum;
1682:D      2                          fid      : file_id_type);
1683:C      2 begin
1684:C      3   with packet_ptr.sclose^ do
1685:C      4   begin
1686:C      5     setup_smh(send_mess_header,
1687:C      6       size_to_close,
1688:C      7       req_close,
1689:C      8       0);
1690:C      9     file_id      := fid;
1691:C      9     directory_password := ' ';
1692:C      9     file_password  := ' ';
1693:C      9     filler5.1     := 0;
1694:C      9     nodeallocate.1 := 0;
1695:C      3   end;
1696:S
1697:C      2   packetout(unum);
1698:C      2   packetin(unum, req_close);
1699:C      2 end;
1700:D -24 1 (*****
1701:S
1702:D      1 procedure copypack(unum      : unitnum;
1703:D      2                          srcfid : file_id_type;
1704:D      3                          srcoff  : integer;
1705:D      4                          destfid : file_id_type;
1706:D      5                          destoff : integer;
1707:D      6                          req     : integer);
1708:C      2 begin
1709:C      3   with packet_ptr.scopy^ do
1710:C      4   begin
1711:C      5     setup_smh(send_mess_header,
1712:C      6       size_to_copy,
1713:C      7       req_copy,
1714:C      8       0);
1715:S      9
1716:C      9     source_file_id      := srcfid;
1717:C      9     source_offset       := srcoff;
1718:C      9     destination_file_id := destfid;
1719:C      9     destination_offset := destoff;
1720:C      9     requested          := req;
1721:C      3   end;
1722:S
1723:C      2   packetout(unum);
1724:C      2   packetin(unum, req_copy);
1725:C      2 end;
1726:D -24 1 (*****
1727:S
1728:D      1 procedure createpack(unum      : unitnum;
1729:D      2                          nfns     : integer;
1730:D      3                          nsaptr    : pnsa;
1731:D      4                          path     : path_start_type;
1732:D      5                          wd       : file_id_type;
1733:D      6                          rtpass   : name_type;
1734:D      7                          nps      : integer;
1735:D      8                          psaptr   : ppsa;
1736:D      9                          ftype    : gang_file_codes;
1737:D      2                          mode    : integer;
```

```
1738:D      2                          maxrec : integer;
1739:D      3                          ext1     : integer;
1740:D      3                          ext2     : integer;
1741:D -16 2                          xaddr    : integer);
1742:C      2 begin
1743:C      3   with packet_ptr.screatefile^ do
1744:C      4   begin
1745:C      5     pnsa(addr(start_name_sets))^ := nsaptr^;
1746:C      5     if nps > 0 then
1747:C      6       ppsa(addr(pnsa(addr(start_name_sets))^[nfns+1]))^ := psaptr^;
1748:S      7
1749:C      3     setup_smh(send_mess_header,
1750:C      4       size_to_create+nfns*36+nps*24,
1751:C      5       req_create,
1752:C      6       0);
1753:S      7
1754:C      3     setup_vnh(volume_name_header, unum);
1755:S      7
1756:C      3     setup_fnh(file_name_header,
1757:C      4       nfns,
1758:C      5       wd, {working directory}
1759:C      6       path,
1760:C      7       rtpass); {root password}
1761:S      7
1762:C      3     file_code      := ftype;
1763:C      3     record_mode     := mode;
1764:S      7
1765:C      3     max_record_size := maxrec;
1766:C      3     first_extent    := ext1;
1767:C      3     contiguous_first_extent.1 := 0; {false}
1768:C      3     secondary_extent := ext2;
1769:C      3     max_file_size    := maxint;
1770:C      3     boot_start_address := xaddr;
1771:C      3     num_protect_code_sets := nps;
1772:C      3     label_included_flag.1 := 0; {false}
1773:C      3   end;
1774:S
1775:C      2   packetout(unum);
1776:C      2   packetin(unum, req_create);
1777:C      2 end;
1778:D -24 1 (*****
1779:S
1780:D      1 procedure createlinkpack(unum      : unitnum;
1781:D      2                          oldnfns  : integer;
1782:D      3                          oldnsaptr : pnsa;
1783:D      4                          oldpath   : path_start_type;
1784:D      5                          oldwd     : file_id_type;
1785:D      6                          oldrtpass : name_type;
1786:D      7                          newnfns  : integer;
1787:D      8                          newnsaptr : pnsa;
1788:D      9                          newpath   : path_start_type;
1789:D      2                          newwd    : file_id_type;
1790:D      3                          newrtpass : name_type;
1791:D      4                          purgeold : boolean);
1792:C      2 begin
1793:C      3   with packet_ptr.screatelink^ do
1794:C      4   begin
1795:C      5     pnsa(addr(start_name_sets))^ := oldnsaptr^;
1796:C      5     pnsa(addr(pnsa(addr(start_name_sets))^[oldnfns+1]))^ := newnsaptr^;
```



```

1788:C      3      setup_smh(send_mess_header,
1799:C      3      size_to_createlink+oldnfns*36+newnfns*36,
1800:C      3      req_createlink,
1801:C      3      0);
1802:S
1803:C      3      setup_vnh(volume_name_header,unum);
1804:C
1805:C      3      setup_fnh(old_file_name_header,
1806:C      3      oldnfns,
1807:C      3      oldwd,      {working directory}
1808:C      3      oldpath,
1809:C      3      oldrtpass);{root password}
1810:S
1811:C      3      setup_fnh(new_file_name_header,
1812:C      3      newnfns,
1813:C      3      newwd,      {working directory}
1814:C      3      newpath,
1815:C      3      newrtpass);{root password}
1816:S
1817:C      3      purge_old_link.i := ord(purgeold);
1818:C      3      end;
1819:C
1820:C      2      packetout(unum);
1821:C      2      packetin(unum,req_createlink);
1822:C      2      end;
1823:D      -24  1  {*****}
1824:S
1825:D      1  procedure exchangeack(unum      : unitnum;
1826:D      2      fid1      : file_id_type;
1827:D      2      fid2      : file_id_type);
1828:C
1829:C      2      begin
1830:C      2      with packet_ptr.sexchange^ do
1831:C      2      begin
1832:C      2      setup_smh(send_mess_header,
1833:C      2      size_to_xchg_open,
1834:C      2      req_xchg_open,
1835:C      2      0);
1836:C
1837:C      2      file_id_1 := fid1;
1838:C      2      file_id_2 := fid2;
1839:C      2      end;
1840:C
1841:C      2      packetout(unum);
1842:C      2      packetin(unum,req_xchg_open);
1843:C      2      end;
1844:D      -24  1  {*****}
1845:D      1  procedure fileinfoack(unum      : unitnum;
1846:D      2      fid      : file_id_type);
1847:C
1848:C      2      begin
1849:C      2      with packet_ptr.sfileinfo^ do
1850:C      2      begin
1851:C      2      setup_smh(send_mess_header,
1852:C      2      size_to_info,
1853:C      2      req_info,
1854:C      2      0);
1855:C
1856:C      2      implicit_unlock.i := 1;
1857:C      2      file_id := fid;
1858:C      2      end;

```

```

1859:S
1860:C      2      packetout(unum);
1861:C      2      packetin(unum,req_info);
1862:C      2      end;
1863:D      -24  1  {*****}
1864:D      1  procedure gangcleanack(unum      : unitnum;
1865:D      2      savewd      : boolean);
1866:C
1867:C      2      begin
1868:C      2      with packet_ptr.sganglean^ do
1869:C      2      begin
1870:C      2      setup_smh(send_mess_header,
1871:C      2      size_to_gang_cleanup,
1872:C      2      req_gang_cleanup,
1873:C      2      0);
1874:C
1875:C      2      keep_protected_directories.i := ord(savewd);
1876:C      2      end;
1877:C
1878:C      2      packetout(unum);
1879:C      2      end;
1880:D      -24  1  {*****}
1881:D      1  procedure lockpack(unum      : unitnum;
1882:D      2      fid      : file_id_type;
1883:D      2      wait      : boolean);
1884:C
1885:C      2      begin
1886:C      2      waitingforlock := wait;
1887:C      2      with packet_ptr.slock^ do
1888:C      2      begin
1889:C      2      setup_smh(send_mess_header,
1890:C      2      size_to_flock,
1891:C      2      req_flock,
1892:C      2      0);
1893:C
1894:C      2      file_id := fid;
1895:C      2      wait_for_lock.i := ord(wait);
1896:C      2      end;
1897:C
1898:C      2      packetout(unum);
1899:C      2      packetin(unum,req_flock);
1900:C      2      end;
1901:D      -24  1  {*****}
1902:D      1  procedure openpack(unum      : unitnum;
1903:D      2      nfns      : integer;
1904:D      2      nsaptr      : pnsa;
1905:D      2      path      : path_start_type;
1906:D      2      wd      : file_id_type;
1907:D      2      rtpass      : name_type;
1908:D      2      share      : integer;
1909:D      2      ppn      : gang_open_type);
1910:C
1911:C      2      begin
1912:C      2      with packet_ptr.sopen^ do
1913:C      2      begin
1914:C      2      pnsa(addr(start_name_sets))^ := nsaptr^;
1915:C
1916:C      2      setup_smh(send_mess_header,
1917:C      2      size_to_open+nfns*36,

```



```
1918:C      3      0);
1919:S
1920:C      3      setup_vnh(volume_name_header,unum);
1921:S
1922:C      3      setup_fnh(file_name_header,
1923:C      3      nfns,
1924:C      3      wd,      (working directory)
1925:C      3      path,
1926:C      3      rtpass);(root password)
1927:S
1928:C      3      filler2      := 0;
1929:C      3      filler3      := 0;
1930:C      3      share_code     := share;
1931:C      3      filler4.id     := 0;
1932:C      3      filler1      := 0;
1933:C      3      open_type     := opn;
1934:C      3      end;
1935:S
1936:C      2      packetout(unum);
1937:C      2      packetin(unum,req_open);
1938:C      2      end;
1939:D      -24      1      (*****
1940:S
1941:D      1      procedure pospack(unum      : unitnum;
1942:D      2      fid      : file_id_type;
1943:D      2      typepos   : position_type;
1944:D      2      boffset   : integer);
1945:C      2      begin
1946:C      2      with packet_ptr.spos^ do
1947:C      2      begin
1948:C      3      setup_smh(send_mess_header,
1949:C      3      size_to_position,
1950:C      3      req_position,
1951:C      3      0);
1952:S
1953:C      3      implicit_unlock.i := 1;
1954:C      3      file_id := fid;
1955:C      3      filler3 := 0;
1956:C      3      type_of_position := typepos;
1957:C      3      byte_offset := boffset;
1958:C      3      end;
1959:S
1960:C      2      packetout(unum);
1961:C      2      packetin(unum,req_position);
1962:C      2      end;
1963:D      -24      1      (*****
1964:S
1965:D      1      procedure purgepack(unum      : unitnum;
1966:D      2      nfns      : integer;
1967:D      2      nsaptr     : pnsa;
1968:D      2      path       : path_start_type;
1969:D      2      wd         : file_id_type;
1970:D      2      rtpass     : name_type);
1971:C      2      begin
1972:C      2      with packet_ptr.spurge^ do
1973:C      3      begin
1974:C      3      pnsa(addr(start_name_sets))^ := nsaptr^;
1975:S
1976:C      3      setup_smh(send_mess_header,
1977:C      3      size_to_purge+nfns*36,
```

```
1978:C      3      req_purgelink,
1979:C      3      0);
1980:S
1981:C      3      setup_vnh(volume_name_header,unum);
1982:S
1983:C      3      setup_fnh(file_name_header,
1984:C      3      nfns,
1985:C      3      wd,      (working directory)
1986:C      3      path,
1987:C      3      rtpass);(root password)
1988:C      3      end;
1989:S
1990:C      2      packetout(unum);
1991:C      2      packetin(unum,req_purgelink);
1992:C      2      end;
1993:D      -24      1      (*****
1994:S
1995:D      1      procedure sendreadpack(unum      : unitnum;
1996:D      2      fid      : file_id_type;
1997:D      2      access     : integer;
1998:D      2      req        : integer;
1999:D      2      off        : integer;
2000:D      2      dat        : anyptr);
2001:C      2      begin
2002:C      2      with packet_ptr.sread^ do
2003:C      3      begin
2004:C      3      setup_smh(send_mess_header,
2005:C      3      size_to_read,
2006:C      3      req_read,
2007:C      3      integer(dat));
2008:S
2009:C      3      implicit_unlock.i := 1;
2010:C      3      file_id := fid;
2011:C      3      access code := access;
2012:C      3      filler3[1] := 0;
2013:C      3      filler3[2] := 0;
2014:C      3      requested := req;
2015:C      3      offset := off;
2016:C      3      end;
2017:S
2018:C      2      packetout(unum);
2019:C      2      end;
2020:D      -24      1      (*****
2021:S
2022:D      1      procedure seteofpack(unum      : unitnum;
2023:D      2      fid      : file_id_type;
2024:D      2      usecurptr   : boolean;
2025:D      2      boffset     : integer);
2026:C      2      begin
2027:C      2      with packet_ptr.sseteof^ do
2028:C      3      begin
2029:C      3      setup_smh(send_mess_header,
2030:C      3      size_to_set_eof,
2031:C      3      req_set_eof,
2032:C      3      0);
2033:S
2034:C      3      implicit_unlock.i := 1;
2035:C      3      file_id := fid;
2036:C      3      use_current_ptr.i := ord(usecurptr);
2037:C      3      byte_offset := boffset;
```



```

2038:C      3      end;
2039:S
2040:C      2      packetout(unum);
2041:C      2      packetin(unum,req_set_eof);
2042:C      2      end;
2043:D      -24 1 (*****
2044:S
2045:D      1      procedure unlockpack(unum      : unitnum;
2046:D      2      fid      : file_id_type);
2047:C      2      begin
2048:C      2      with packet_ptr.sunlock^ do
2049:C      3      begin
2050:C      3      setup_smh(send_mess_header,
2051:C      3      size_to_funlock,
2052:C      3      req_funlock,
2053:C      3      0);
2054:S
2055:C      3      file_id      := fid;
2056:C      3      explicit_unlock.i := ord(true);
2057:C      3      end;
2058:S
2059:C      2      packetout(unum);
2060:C      2      packetin(unum,req_funlock);
2061:C      2      end;
2062:D      -24 1 (*****
2063:S
2064:D      1      procedure volpack(unum : unitnum);
2065:C      2      begin
2066:C      2      with packet_ptr.svol^ do
2067:C      3      begin
2068:C      3      setup_smh(send_mess_header,
2069:C      3      size_to_volstatus,
2070:C      3      req_volstatus,
2071:C      3      0);
2072:S
2073:C      3      setup_vnh(volume_name_header,unum);
2074:C      3      end;
2075:S
2076:C      2      packetout(unum);
2077:C      2      packetin(unum,req_volstatus);
2078:C      2      end;
2079:D      -24 1 (*****
2080:S
2081:D      1      procedure sendwritepack(unum      : unitnum;
2082:D      2      fid      : file_id_type;
2083:D      2      access   : integer;
2084:D      2      req      : integer;
2085:D      2      off      : integer;
2086:D      2      dat      : anyptr);
2087:C      2      begin
2088:C      2      with packet_ptr.swrite^ do
2089:C      3      begin
2090:C      3      setup_smh(send_mess_header,
2091:C      3      size_to_write + req,
2092:C      3      req_write,
2093:C      3      integer(dat));
2094:S
2095:C      3      implicit_unlock.i := 1;
2096:C      3      file_id      := fid;
2097:C      3      access_code   := access;

```

```

2098:C      3      filler3[1] := 0;
2099:C      3      filler3[2] := 0;
2100:C      3      requested := req;
2101:C      3      offset      := off;
2102:C      3      filler8.i := 0;
2103:C      3      flush_buffer.i := 1;
2104:C      3      moveleft(charptr(dat)^,data,req);
2105:C      3      end;
2106:S
2107:C      2      packetout(unum);
2108:C      2      end;
2109:D      -24 1 (*****
2110:S
2111:C      1      end(module srm).
2112:S
2113:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

SWVOL

Description

SWVOL prompts for disc changes during stream files. Input is taken from the keyboard unit rather than the stream file.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```
1:D      0 $copyright 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'$
2:D      0 $MODCAL$
3:D      0 program switchvols(input, output);
4:D      1 import sysglobals;
5:D      1 var
6:D      -1 1 ch: char;
7:D      -36 1 vol_name: string[34];
8:D      -40 1 unit_number: integer;
9:S
10:C      1 begin
11:C      1 readln(vol_name);
12:C      1 if vol_name = '*' then
13:C      2 begin vol_name := syvid; unit_number := sysunit; end
14:C      2 else readln(unit_number);
15:S
16:C      1 page;
17:C      1 writeln; writeln; writeln;
18:C      1 if unit_number > 0 then writeln('Please put ', vol_name,
19:C      2 ' in unit #', unit_number:1)
20:C      2 else writeln('Please make sure ', vol_name, ' is on line');
21:C      1 write (' and press the X key..');
22:C      1 with untable^[fbbp(gfiles[2])^.funit] do
23:C      2 repeat
24:C      3 call(tm, fbbp(gfiles[2]), readbytes, ch, 1, 0);
25:C      3 until (ch='x') or (ch='X');
26:C      1 writeln(ch);
27:C      1 end.
```

No errors. No warnings.

***** Nonstandard language features enabled *****

SYSDEVS

Description

SYSDEVS provides declarations and “hooks” for the keyboard, the CRT, timers, interrupts, and typeahead buffer declarations and functionality.

Requirements

SYSGLOBALS.

Notes

SYSDEVS is described in Chapter 14 of the *Pascal 3.0 Procedure Library* manual.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1984.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S     RESTRICTED RIGHTS LEGEND
11:S
12:S     Use, duplication, or disclosure by the Government
13:S     is subject to restrictions as set forth in
14:S     paragraph (b) (3) (B) of the Rights in Technical
15:S     Data and Computer Software clause in
16:S     DAR 7-104.9(a).
17:S
18:S     HEWLETT-PACKARD COMPANY
19:D     0 Fort Collins, Colorado      *)
20:S
21:D     0 $SYSPROGS
22:D     0 $heap_dispose off$
23:D     0 $iocheck off$
24:D     0 $range off$ $ovflcheck off$
25:D     0 $debug off$
26:D     0 $stackcheck off$
27:S
28:D     0 PROGRAM SYSDEVICES;
29:D     1 MODULE SYSDEVS;
30:S
31:D     1 $SEARCH 'INITLOAD'$
32:S
33:D     1 IMPORT SYSGLOBALS;
34:D     1 EXPORT
35:D     1 (* DUMMY DECLARATIONS *****
36:D     1 TYPE
37:D     1 KBDHOOKTYPE = PROCEDURE(VAR STATBYTE, DATABYTE: BYTE;
38:D     1                               VAR DOIT: BOOLEAN);
39:D     1 OUT2TYPE = PROCEDURE(VALUE1, VALUE2: BYTE);
40:D     1 REQUESTTYPE = PROCEDURE(CMD: BYTE; VAR VALUE: BYTE);
41:D     1 BOOLPROC = PROCEDURE(B: BOOLEAN);
42:S
43:D     1 (* CRT *****
44:D     1 {***** THIS SECTION HAS HARD OFFSET REFERENCES *****}
45:D     1 IN MODULES CRT8 (ASSY FILE GASSM)
46:D     1 TYPE
47:D     1 CRTWORD = RECORD CASE INTEGER OF
48:D     1 1:(HIGHLIGHTBYTE, CHARACTER: CHAR);
49:D     1 2:(WHOLEWORD: SHORTINT);
50:D     1 END;
51:D     1 CRTLLOPS = (CLLPUT, CLLSHIFTL, CLLSHIFTR, CLLCLEAR, CLLDISPLAY, PUTSTATUS);
52:D     1 CRTLLTYPE = PROCEDURE(OP: CRTLLOPS; ANYVAR POSITION: INTEGER; C: CHAR);
53:D     1 DBCRTOPS = (DBINFO, DBEXCG, DBGOTOXY, DBPUT, DBINIT, DBCLEAR, DBCLINE, DBSCROLLUP,
54:D     1 DBSCROLLDN, DBSCROLLL, DBSCROLLR, DBHIGH);
55:D     1 DBCINFO = RECORD
56:D     1 SAVEAREA : WINDOWP;
57:D     1 SAVE SIZE : INTEGER;
58:D     1 DCURSADDR : INTEGER;
59:D     1 XMIN, XMAX, YMIN, YMAX : SHORTINT;
60:D     1 CURSX, CURSY : SHORTINT;

```

```

61:D     1 C : CHAR;
62:D     1 AREASDBCRT : BOOLEAN;
63:D     1 END;
64:D     1 DBCRTTYPE = PROCEDURE(OP: DBCRTOPS; VAR DBCRT: DBCINFO);
65:S
66:D     1 crtconsttype = packed array [0..11] of byte;
67:S
68:D     1 crtrec = packed record
69:D     1 nobreak, stupid, slowterm, hasxycrt,
70:D     1 haslcrt(built in crt), hasclock,
71:D     1 canupscroll, candownscroll : boolean;
72:D     1 end;
73:S
74:D     1 b3 = packed array[0..8] of boolean;
75:D     1 b14 = packed array[0..13] of boolean;
76:D     1 crtrec = packed record
77:D     1 rlf, ndfs, eraseeol,
78:D     1 eraseeos, home,
79:D     1 escape : char;
80:D     1 backspace : char;
81:D     1 fillcount : 0..255;
82:D     1 clearscreen,
83:D     1 clearline : char;
84:D     1 prefixed : b9
85:D     1 end;
86:S
87:D     1 crtirc = packed record
88:D     1 width, height : shortint;
89:D     1 crtmemaddr, crtcontroladdr,
90:D     1 keybufferaddr, progstateinfoaddr: integer;
91:D     1 keybuffer size: shortint;
92:D     1 crtcon : crtconsttype;
93:D     1 right, left, down, up: char;
94:D     1 badch, char del, stop,
95:D     1 break, flush, eof : char;
96:D     1 altmode, linedel : char;
97:D     1 backspace,
98:D     1 etx, prefix : char;
99:D     1 prefixed : b14;
100:D     1 cursormask : integer;
101:D     1 spare : integer;
102:D     1 end;
103:S
104:D     1 environ = record
105:D     1 miscinfo: crtrec;
106:D     1 crttype: integer;
107:D     1 crtctrl: crtrec;
108:D     1 crtinfo: crtirc;
109:D     1 end;
110:S
111:D     1 environptr = ^environ;
112:S
113:D     1 crt kinds = (NOCRT, ALPHATYPE, BITMAPTYPE, SPECIALCRT1, SPECIALCRT2);
114:S
115:D     1 VAR
116:D     -4 1 SYSCOM: ENVIRONPTR;
117:D     -4 1 ALPHASTATE['ALPHAFLAG'] : BOOLEAN;
118:D     -4 1 GRAPHICSTATE['GRAPHICSFLAG'] : BOOLEAN;
119:D     -12 1 CRTIOHOOK : ANTYPE;
120:D     -20 1 TOGGLEALPHAHOOK : PROCEDURE;

```



```
121:D -28 1 TOGGLEGRAPHICSHOOK : PROCEDURE;
122:D -36 1 DUMPALPHAHOOK : PROCEDURE;
123:D -44 1 DUMPGRAPHICSHOOK : PROCEDURE;
124:D -52 1 UPDATECURSORHOOK : PROCEDURE;
125:D -60 1 CRTINITHOOK : PROCEDURE;
126:D -68 1 CRTLLHDDK : CRTLLTYPE;
127:D -76 1 DBCRTHDDK : DBCRTTYPE;
128:D -78 1 XPOS : SHORTINT; { CURSOR X POSITION }
129:D -80 1 YPOS : SHORTINT; { CURSOR Y POSITION }
130:D -82 1 CURRENTCRT : CRTKINDS; { ACTIVE ALPHA DRIVER TYPE }
131:D -86 1 BITMAPADDR : INTEGER; { ADDRESS OF BITMAP CONTROL SPACE }
132:D -90 1 FRAMEADDR : INTEGER; { ADDRESS OF BITMAP FRAME BUFFER }
133:D -92 1 REPLREGCOPY : SHORTINT; { REGISTER COPIES FOR BITMAP DISPLAY }
134:D -94 1 WINDOWREGCOPY : SHORTINT; { MUST BE IN GLOBALS BECAUSE REGISTERS }
135:D -96 1 WRITEREGCOPY : SHORTINT; { ARE NOT READABLE -- MAY BE UNDEFINED }
136:S
137:D -96 1 { * KEYBOARD ***** }
138:D -96 1 CONST
139:D -96 1 KBD_ENABLE = 0; KBD_DISABLE = 1;
140:D -96 1 SET_AUTO_DELAY = 2; SET_AUTO_REPEAT = 3;
141:D -96 1 GET_AUTO_DELAY = 4; GET_AUTO_REPEAT = 5;
142:D -96 1 SET_KBDTYPE = 6; SET_KBDLANG = 7;
143:D -96 1 TYPE
144:D -96 1 STRING80PTR = ^STRING80;
145:D -96 1 KEYBOARDTYPE = (NO_KBD, FINISH_KBD, BELGIAN_KBD, CON_ENG_KBD, CON_FR_KBD,
146:D -96 1 LANGTYPE = (NO_KBD, FINISH_KBD, BELGIAN_KBD, CON_ENG_KBD, CON_FR_KBD,
147:D -96 1 NORWEGIAN_KBD, DANISH_KBD, DUTCH_KBD, SWISS_GR_KBD, SWISS_FR_KBD,
148:D -96 1 SPANISH_EUR_KBD, SPANISH_LATIN_KBD, UK_KBD, ITALIAN_KBD,
149:D -96 1 FRENCH_KBD, GERMAN_KBD, SWEDISH_KBD, SPANISH_KBD,
150:D -96 1 KATAKANA_KBD, US_KBD, ROMAN_KBD, NS1_KBD, NS2_KBD, NS3_KBD);
151:D -96 1 MENUTYPE = (M_NONE, M_SYSNORM, M_SYSSHIFT, M_U1, M_U2, M_U3, M_U4);
152:D -96 1 VAR
153:D -104 1 KBREQHOOK : REQUEST1TYPE;
154:D -112 1 KBDIOHOOK : AMTYPE;
155:D -120 1 KBDISRHOOK : KBDHOOKTYPE;
156:D -128 1 KBDPOLHOOK : BOOLPROC;
157:D -130 1 KBDTYPE : KEYBOARDTYPE;
158:D -132 1 KBCONFIG : BYTE; { KEYBOARD CONFIGURATION JUMPER }
159:D -134 1 KBDLANG : LANGTYPE;
160:D -138 1 SYSMENU : STRING80PTR;
161:D -142 1 SYSMENUSHIFT : STRING80PTR;
162:D -144 1 MENUSTATE : MENUTYPE;
163:S
164:D -144 1 { * ENABLE / DISABLE ***** }
165:D -144 1 CONST
166:D -144 1 KSDMASK=1; RESETMASK=2; TIMERMASK=4; PSIMASK=8; FHIMASK=16;
167:D -144 1 VAR
168:D -152 1 MASKOPSHOOK : OUT2TYPE; { ENABLE, DISABLE }
169:S
170:D -152 1 { * BEEPER ***** }
171:D -152 1 VAR
172:D -160 1 BEEPERHOOK : OUT2TYPE;
173:D -164 1 BFREQUENCY, BOURATION: BYTE;
174:S
175:D -164 1 { * RPG ***** }
176:D -164 1 CONST
177:D -164 1 RPG_ENABLE = 0; RPG_DISABLE = 1;
178:D -164 1 SET_RPG_RATE = 2; GET_RPG_RATE = 3;
179:D -164 1 VAR
180:D -172 1 RPGREQHOOK : REQUEST1TYPE;
```

```
181:D -180 1 RPGISRHOOK : KBDHOOKTYPE;
182:S
183:D -180 1 { * BATTERY ***** }
184:D -180 1 TYPE
185:D -180 1 BATCMDTYPE = PROCEDURE(CMD: BYTE; NUMDATA: INTEGER;
186:D -180 1 B1, B2, B3, B4, B5: BYTE);
187:D -180 1 BATREADTYPE = PROCEDURE(VAR DATA: BYTE);
188:D -180 1 VAR
189:D -180 1 BATTERYPRESENT(-S63): BOOLEAN;
190:D -180 1 BATCMDHOOK : BATCMDTYPE;
191:D -196 1 BATREADHOOK : BATREADTYPE;
192:S
193:D -186 1 { * CLOCK ***** }
194:D -196 1 TYPE
195:D -196 1 RTCTIME = PACKED RECORD
196:D -196 1 PACKEDTIME, PACKEODATE: INTEGER;
197:D -196 1 END
198:D -196 1 CLDCKFUNC = (CGETDATE, CGETTIME, CSETDATE, CSETTIME);
199:D -196 1 CLOCKOP = (CGET, CSET);
200:D -186 1 CLOCKDATA = RECORD
201:D -186 1 CASE 800LEAF OF
202:D -196 1 TRUE : (TIMETYPE: TIMERECD);
203:D -196 1 FALSE : (DATETYPE: DATERECD);
204:D -196 1 END
205:D -196 1 CLOCKREQTYPE = PROCEDURE(CMD: CLOCKFUNC; ANYVAR DATA: CLOCKDATA);
206:D -196 1 CLOCKIOTYPE = PROCEDURE(CMD: CLOCKOP; VAR DATA: RTCTIME);
207:D -196 1 VAR
208:D -204 1 CLDCKREQHOOK : CLDCKREQTYPE; { CLDCK MODULE INTERFACE }
209:D -212 1 CLOCKIOHOOK : CLOCKIOTYPE; { CARO DRIVER INTERFACE }
210:S
211:D -212 1 { * TIMER ***** }
212:D -212 1 TYPE
213:D -212 1 TIMERTYPES = (CYCLICT, PERIDICT, DELAYT, MATCHT);
214:D -212 1 TIMEROPTYPE = (SETT, READT, GETTINFO);
215:D -212 1 TIMERDATA = RECORD
216:D -212 1 CASE INTEGER OF
217:D -212 1 0 : (COUNT: INTEGER);
218:D -212 1 1 : (MATCH: TIMERECD);
219:D -212 1 2 : (RESOLUTION, RANGE: INTEGER);
220:D -212 1 END
221:D -212 1 TIMERIDTYPE = PROCEDURE(TIMER: TIMERTYPES; OP: TIMEROPTYPE; VAR TD: TIMERDATA);
222:D -212 1 VAR
223:D -220 1 TIMERIOHOOK : TIMERIOTYPE;
224:D -228 1 TIMERISRHOOK : KBDHOOKTYPE;
225:S
226:D -228 1 { * KEYBUFFER ***** }
227:D -228 1 CONST
228:D -228 1 KMAXBUFSIZE = 255;
229:D -228 1 TYPE
230:D -228 1 TYPE
231:S
232:D -228 1 KOPTYPE = (KGETCHAR, KAPPEND, KNOADVANCE, KCLEAR, KDISPLAY,
233:D -228 1 KGETLAST, KPUTFIRST);
234:D -228 1 KBUFTYPE = PACKED ARRAY[0..KMAXBUFSIZE] OF CHAR;
235:D -228 1 KBUFPTTR = ^KBUFTYPE;
236:D -228 1 KBUFRECPTTR = ^KBUFREC;
237:D -228 1 KBUFREC = RECORD
238:D -228 1 ECHO: BOOLEAN;
239:D -228 1 NON_CHAR: CHAR;
240:D -228 1 MAXSIZE, SIZE, INP, OUTP: INTEGER;
```



```

241:D -228 1      BUFFER: KBUFPTR;
242:D -228 1      END;
243:S
244:D -228 1 VAR
245:D -232 1      KEYBUFFER: KBUFRECPT;
246:D -240 1      KBDWAITHOOK: PROCEDURE;
247:D -248 1      KBDRELEASEHOOK: PROCEDURE;
248:D -256 1      STATUSLINE: PACKED ARRAY[0..7] OF CHAR;
249:D -256 1      { 0 s or f = STEP/FLASH IN PROGRESS (WAITING FOR TRAP #0) }
250:D -256 1      { 1..S last executed/current line number }
251:D -256 1      { 6 S=SYSTEM U=USER DEFINITION FOR ITF SOFT KEYS }
252:D -256 1      { BLANK FOR NON ITF KEYBOARDS }
253:D -256 1      { 7 RUNLIGHT }
254:S
255:D -256 1 (* KEY TRANSLATION SERVICES ***** )
256:D -256 1 TYPE
257:D -256 1 KEYTRANSTYPE = (KPASSTHRU, KSHIFT_EXT, KPASS_EXT);
258:D -256 1 KEYTYPE = (ALPHA_KEY, NONADV_KEY, SPECIAL_KEY, IGNORED_KEY, NONA_ALPHA_KEY);
259:D -256 1 { ADDED NONA_ALPHA_KEY 5/9/84 RQ/SFB }
260:S
261:D -256 1 LANGCOMREC = RECORD
262:D -256 1     STATUS: BYTE;
263:D -256 1     DATA: BYTE;
264:D -256 1     KEY: CHAR;
265:D -256 1     RESULT: KEYTYPE;
266:D -256 1     SHIFT, CONTROL, EXTENSION: BOOLEAN;
267:D -256 1 END;
268:D -256 1 LANGKEYREC = RECORD
269:D -256 1     NO_CAPSLOCK: BOOLEAN;
270:D -256 1     NO_SHIFT: BOOLEAN;
271:D -256 1     NO_CONTROL: BOOLEAN;
272:D -256 1     NO_EXTENSION: BOOLEAN;
273:D -256 1     KEYCLASS: KEYTYPE;
274:D -256 1     KEYS: ARRAY[BOOLEAN] OF CHAR;
275:D -256 1 END;
276:D -256 1 LANGRECORD = RECORD
277:D -256 1     CAN_NONADV: BOOLEAN;
278:D -256 1     LANGCODE: LANGCODE;
279:D -256 1     SEMANTICS: PROCEDURE;
280:D -256 1     KEYTABLE: ARRAY[0..127] OF LANGKEYREC;
281:D -256 1 END;
282:D -256 1 LANGPTR = ^LANGRECORD;
283:D -256 1 VAR
284:D -268 1     LANGCOM: LANGCOMREC;
285:D -276 1     LANGTABLE: ARRAY[0..1] OF LANGPTR;
286:D -278 1     LANGINDEX: 0..1;
287:D -286 1     KBDTRANSHOOK: KBDHOOKTYPE;
288:D -288 1     TRANSMODE: KEYTRANSTYPE;
289:D -291 1     KBDYSMMODE, KBDALTLOCK, KBDCAPSLOCK: BOOLEAN;
290:S
291:D -291 1 (* HPHIL ***** )
292:D -291 1 { MOVED INTO SYSDEVS 4/6/84 SFB }
293:D -291 1 CONST
294:D -291 1     le_configured = hex('80');
295:D -291 1     le_error = hex('81');
296:D -291 1     le_timeout = hex('82');
297:D -291 1     le_loopdown = hex('84');
298:S
299:D -291 1     lmaxdevices = 7;
300:S

```

```

301:D -291 1 TYPE
302:D -291 1     loopdvrop = (datastarting, dataended, resetdevice);
303:D -291 1     loopdvrrproc = PROCEDURE (OP: loopdvrop);
304:S
305:D -291 1 { HPHILOP DEFINED AS NEW TYPE 4/6/84 SFB }
306:D -291 1 HPHILOP = (RAWSHIFTOP, NORMSHIFTOP, CHECKLOOPOP, CONFIGUREOP, LCOMMANDOP);
307:D -291 1 { 5 PROCEDURES HOOKED AS TYPE HPHILCMDPROC 4/6/84 SFB }
308:D -291 1 HPHILCMDPROC = PROCEDURE (OP: HPHILOP);
309:S
310:S
311:D -291 1 descrip = packed record { DEVICE DESCRIBE RECORD }
312:D -291 1     case boolean of
313:D -291 1     true: (id: byte;
314:D -291 1         twosets: boolean;
315:D -291 1         abscoords: boolean;
316:D -291 1         sizel6: boolean;
317:D -291 1         hasprompts: boolean;
318:D -291 1         reserved: 0..3;
319:D -291 1         numaxes: 0..3;
320:D -291 1         counts: shortint;
321:D -291 1         maxcountx: shortint;
322:D -291 1         maxcounty: shortint;
323:D -291 1         maxcountz: shortint;
324:D -291 1         nprompts: 0..7;
325:D -291 1         nbuttons: 0..7;
326:D -291 1     false: (darray: array[1..11] of char);
327:D -291 1     end;
328:S
329:D -291 1 devicerec = record
330:D -291 1     devstate: integer;
331:D -291 1     descrip: descrip;
332:D -291 1     opsvproc: loopdvrrproc;
333:D -291 1     dataproc: kbdhooktype;
334:D -291 1     end;
335:S
336:D -291 1 loopdvrrptr = ^loopdriverrec;
337:D -291 1 loopdriverrec = record
338:D -291 1     lowid, highid, daddr: byte;
339:D -291 1     opsvproc: loopdvrrproc;
340:D -291 1     dataproc: kbdhooktype;
341:D -291 1     next: loopdvrrptr;
342:D -291 1     end;
343:S
344:D -291 1 LOOPCONTROLREC = RECORD { REDEFINED AS RECORD - 4/6/84 SFB }
345:D -291 1     rawmode: boolean;
346:D -291 1     loopdevices: array[1..lmaxdevices] of devicerec;
347:D -291 1     loopdevice: 1..lmaxdevices;
348:D -291 1     loopcmd: byte; { last loop command sent }
349:D -291 1     loopdata: byte; { data byte in / out }
350:D -291 1     looperror: boolean; { error occurred on last operation }
351:D -291 1     loopinconfig: boolean; { now doing reconfigure }
352:D -291 1     loopcmddone: boolean; { last sent command is done }
353:D -291 1     loopisok: boolean; { loop is configured }
354:D -291 1     loopdevreading: boolean; { reading poll data } { 3.0 BUG #39 3/17/84 }
355:D -291 1 END;
356:S
357:D -291 1 VAR
358:D -291 1     loopdriverlist: loopdvrrptr;
359:D -291 1     LOOPCONTROL: ^LOOPCONTROLREC; { 4/6/84 SFB }
360:D -300 1

```



```

361:D -308 1 HPHILCMDHOOK : HPHILCMDPROC; {4/6/84 5FB}
362:S
363:D -308 1 {-----}
364:D -308 1 PROCEDURE SYSDEV_INIT;
365:D -308 1 { * BEEPER *****}
366:D -308 1 PROCEDURE BEEP;
367:D -308 1 PROCEDURE BEEPER(FREQUENCY,DURATION:BYTE);
368:D -308 1 { * RPG *****}
369:D -308 1 PROCEDURE SETRPGRATE(RATE : BYTE);
370:D -308 1 { * KEYBOARD *****}
371:D -308 1 PROCEDURE KBDSETUP(CMD,VALUE:BYTE);
372:D -308 1 PROCEDURE KBDIO(FP: FIBP; REQUEST: AMREQUESTTYPE;
373:D -308 2 ANYVAR BUFFER: WINDOW; BUFSIZE,POSITION: INTEGER);
374:D -308 1 procedure lockedaction(a: action);
375:D -308 1 { * CRT *****}
376:D -308 1 PROCEDURE CRTIO(FP: FIBP; REQUEST: AMREQUESTTYPE;
377:D -308 2 ANYVAR BUFFER: WINDOW; BUFSIZE,POSITION: INTEGER);
378:D -308 1 PROCEDURE DUMMYCRTLL(DP: CRTLLOPS; ANYVAR POSITION: INTEGER; C: CHAR);
379:D -308 1 { * BATTERY *****}
380:D -308 1 PROCEDURE BATCOMMAND(CMD: BYTE; NUMDATA: INTEGER; B1, B2, B3, B4, B5: BYTE);
381:D -308 1 FUNCTION BATBYTE RECEIVED: BYTE;
382:D -308 1 { * CLOCK *****}
383:D -308 1 function sysclock: integer; {centiseconds from midnight}
384:D -308 1 procedure sysdate (var thedate: daterec);
385:D -308 1 procedure systime (var thetime: timerec);
386:D -308 1 procedure setsysdate ( thedate: daterec);
387:D -308 1 procedure setsystime ( thetime: timerec);
388:D -308 1 { * KEYBUFFER *****}
389:D -308 1 PROCEDURE KEYBUFOPS(OP: KOPTYPE; VAR C: CHAR);
390:D -308 1 { * STATUSLINE *****}
391:D -308 1 PROCEDURE SETSTATUS(N: INTEGER; C: CHAR);
392:D -308 1 FUNCTION RUNLIGHT: CHAR;
393:D -308 1 PROCEDURE SETRUNLIGHT(C: CHAR);
394:S
395:S -308 1 IMPLEMENT
396:S
397:D -308 1 { * GENERAL PURPOSE DUMMY PROCESSES *****}
398:D -308 1 PROCEDURE DUMMYPROC;
399:D -308 2 BEGIN END;
400:D -308 1 PROCEDURE DUMMYOUT2(VALUE1, VALUE2 : BYTE);
401:D -308 2 BEGIN END;
402:D -308 1 PROCEDURE DUMMYKBD(VAR STATBYTE,DATABYTE : BYTE;
403:D -308 2 VAR DOIT: BOOLEAN);
404:D -308 2 BEGIN END;
405:D -308 1 PROCEDURE DUMMYREQ1(CMD : BYTE; VAR VALUE: BYTE);
406:D -308 2 BEGIN END;
407:D -308 1 PROCEDURE DUMMYTH(FP: FIBP; REQUEST: AMREQUESTTYPE;
408:D -308 2 ANYVAR BUFFER: WINDOW; BUFSIZE,POSITION: INTEGER);
409:D -308 2 BEGIN END;
410:D -308 1 PROCEDURE DUMMYBOOLPROC(B: BOOLEAN);
411:D -308 2 BEGIN END;
412:D -308 1 { * BEEPER *****}
413:D -308 1 PROCEDURE BEEP;
414:D -308 2 BEGIN CALL(BEEPERHOOK,BFREQUENCY,BDURATION);
415:D -308 2 END;
416:D -308 1 PROCEDURE BEEPER(FREQUENCY,DURATION:BYTE);
417:D -308 2 BEGIN CALL(BEEPERHOOK,FREQUENCY,DURATION);
418:D -308 2 END;
419:S
420:D -308 1 { * RPG *****}

```

```

421:D 1 PROCEDURE SETRPGRATE(RATE:BYTE);
422:D -2 2 VAR IRATE : BYTE;
423:D 2 BEGIN
424:D 2 IRATE := RATE; CALL(RPGREQHOOK,SET_RPG_RATE,IRATE);
425:D 2 END;
426:S
427:D -308 1 { * KEYBOARD *****}
428:D -308 1 PROCEDURE KBDSETUP(CMD,VALUE : BYTE);
429:D -2 2 VAR IVALUE : BYTE;
430:D 2 BEGIN IVALUE := VALUE; CALL(KBREQHOOK,CMD,IVALUE);
431:D 2 END;
432:D 1 PROCEDURE KBDIO(FP: FIBP; REQUEST: AMREQUESTTYPE;
433:D 2 ANYVAR BUFFER: WINDOW; BUFSIZE,POSITION: INTEGER);
434:D 2 BEGIN CALL(KBDIOHOOK,FP,REQUEST,BUFFER,BUFSIZE,POSITION); END;
435:S
436:D -308 1 { * BATTERY *****}
437:D -308 1 PROCEDURE BATCOMMAND(CMD:BYTE; NUMDATA:INTEGER; B1, B2, B3, B4, B5: BYTE);
438:D 2 BEGIN CALL(BATCMDHOOK, CMD, NUMDATA, B1, B2, B3, B4, B5);
439:D 2 END;
440:D 1 FUNCTION BATBYTE RECEIVED: BYTE;
441:D -2 2 VAR DATA : BYTE;
442:D 2 BEGIN CALL(BATRDHOOK,DATA); BATBYTE RECEIVED := DATA;
443:D 2 END;
444:S
445:D 1 PROCEDURE DUMMYBATCMD(CMD:BYTE; NUMDATA:INTEGER; B1, B2, B3, B4, B5: BYTE);
446:D 2 BEGIN END;
447:D 1 PROCEDURE DUMMYBATREAD(VAR DATA:BYTE);
448:D 2 BEGIN END;
449:S
450:D -308 1 { * CLOCK *****}
451:D -308 1 PROCEDURE DUMMYCLOCKSYS(VAR STIME:INTEGER);
452:D 2 BEGIN END;
453:D 1 PROCEDURE DUMMYCLOCKREQ(CMD:CLDCKFUNC; ANYVAR DATA:CLOCKDATA);
454:D 2 BEGIN END;
455:D 1 PROCEDURE DUMMYCLOCKIO(CMD:CLOCKOP ; VAR DATA:RTCTIME);
456:D 2 BEGIN END;
457:D 1 function sysclock: integer;
458:D -8 2 var ltime: rctime;
459:D 2 begin
460:D 2 CALL(CLOCKIDHOOK,CGET,LTIME); sysclock := ltime.packedtime;
461:D 2 end;
462:S
463:D 1 procedure sysdate (var thedate: daterec);
464:D 2 BEGIN CALL(CLOCKREQHOOK,CGETDATE,THEDATE);
465:D 2 END;
466:D 1 procedure systime (var thetime: timerec);
467:D 2 BEGIN CALL(CLOCKREQHOOK,CGETTIME,THETIME);
468:D 2 END;
469:D 1 procedure setsysdate ( thedate: daterec);
470:D -2 2 VAR D: DATERECD;
471:D 2 BEGIN D:=THEDATE, CALL(CLDCKREQHOOK,CSETDATE,D);
472:D 2 END;
473:D 1 procedure setsystime ( thetime: timerec);
474:D -4 2 VAR T: TIMERECD;
475:D 2 BEGIN T:=THETIME, CALL(CLOCKREQHOOK,CSETTIME,T);
476:D 2 END;
477:D 1 PROCEDURE DUMMYTIMERIO(TIMER: TIMERTYPES; OP: TIMEROPTYPE; VAR TD: TIMERDATA);
478:D 3 BEGIN IF OP=READ THEN TO.COUNT:=1
479:D 3 ELSE
480:D 3 IF OP=GETINFO THEN TO.RESOLUTION:=0;

```



```

481:C      2      END;
482:S
483:D      -308  1  (* CRT *****
484:D      1      PROCEDURE CRTID (P: FIBP; REQUEST: AREQUESTTYPE;
485:D      2      PROCEDURE CRTID (P: FIBP; REQUEST: AREQUESTTYPE;
486:C      2      BEGIN CALL (CRTIDHOOK, P, REQUEST, BUFFER, BUFSIZE, POSITION); END;
487:D      1      PROCEDURE DUMMYCRTLL (OP: CRTLOPS; ANYVAR POSITION: INTEGER; C: CHAR);
488:C      2      BEGIN END;
489:D      1      PROCEDURE DUMMYDBCRT (OP: DBCRTOPS; VAR DBCRT: DBCINFO);
490:C      2      BEGIN END;
491:S
492:D      1      procedure lockedaction(a: action);
493:D      2      label 1;
494:D      -4      var i: integer;
495:C      2      begin
496:C      2      if locklevel = 0 then call(a)
497:C      3      else
498:C      3      begin
499:C      3      i := actionspending;
500:C      3      while i > 0 do if deferredaction[i] = a then goto 1 else i := i - 1;
501:C      3      if actionspending = 10 then beep
502:C      4      else
503:C      4      begin
504:C      4      actionspending := actionspending + 1;
505:C      4      deferredaction[actionspending] := a;
506:C      4      end;
507:C      3      end;
508:C      2      i
509:C      2      end;
510:S
511:D      -308  1  (* KEYBUFFER *****
512:D      1      PROCEDURE KEYBUFOPS (OP: KOPTYPE; VAR C: CHAR);
513:D      2      VAR
514:D      -8      POSITION, TEMP1: INTEGER;
515:S
516:D      2      PROCEDURE ADVANCE (VAR P: INTEGER);
517:C      3      BEGIN P := (P+1) MOD KEYBUFFER^.MAXSIZE; END;
518:S
519:D      2      PROCEDURE BACKUP (VAR P: INTEGER);
520:C      3      BEGIN IF P = 0 THEN P := KEYBUFFER^.MAXSIZE-1 ELSE P := P-1;
521:C      3      END;
522:S
523:D      2      BEGIN { KEYBUFOPS }
524:C      2      WITH KEYBUFFER^ DO
525:C      3      CASE OP OF
526:C      4      KGETCHAR:
527:C      4      BEGIN { MUST NOT CALL IF SIZE=0 }
528:C      4      IF ECHO THEN CALL (CRTLLHOOK, CLLSHIFTL, POSITION, ' ');
529:C      4      C := BUFFER^[OUTP]; SIZE := SIZE-1; ADVANCE (OUTP);
530:C      4      END;
531:C      4      KAPPEND, KNONADVANCE:
532:C      4      BEGIN { MUST NOT CALL IF SIZE >= MAXSIZE }
533:C      4      IF ECHO THEN CALL (CRTLLHOOK, CLLPUT, SIZE, C);
534:C      4      IF OP = KAPPEND THEN
535:C      5      BEGIN
536:C      5      SIZE := SIZE+1; BUFFER^[INP] := C; ADVANCE (INP); NON_CHAR := ' ';
537:C      5      END;
538:C      5      ELSE NON_CHAR := C;
539:C      4      END;
540:C      4      KCLEAR:

```

```

541:C      4      BEGIN
542:C      4      IF ECHO THEN CALL (CRTLLHOOK, CLLCLEAR, POSITION, ' ');
543:C      4      SIZE := 0; INP := OUTP; NON_CHAR := ' ';
544:C      4      END;
545:C      4      KDISPLAY:
546:C      4      IF ECHO THEN
547:C      5      BEGIN
548:C      5      CALL (CRTLLHOOK, CLLCLEAR, POSITION, ' ');
549:C      5      POSITION := 0; TEMP1 := OUTP;
550:C      5      WHILE POSITION < SIZE DO
551:C      6      BEGIN
552:C      6      CALL (CRTLLHOOK, CLLPUT, POSITION, BUFFER^[TEMP1]);
553:C      6      ADVANCE (TEMP1); POSITION := POSITION+1;
554:C      6      END;
555:C      5      IF NON_CHAR <> ' ' THEN
556:C      6      CALL (CRTLLHOOK, CLLPUT, POSITION, NON_CHAR);
557:C      5      END;
558:C      4      KGETLAST:
559:C      4      BEGIN
560:C      4      IF ECHO AND (NON_CHAR <> ' ') THEN CALL (CRTLLHOOK, CLLPUT, SIZE, ' ');
561:C      4      IF SIZE > 0 THEN
562:C      5      BEGIN
563:C      5      BACKUP (INP); C := BUFFER^[INP]; SIZE := SIZE-1; NON_CHAR := ' ';
564:C      5      IF ECHO THEN CALL (CRTLLHOOK, CLLPUT, SIZE, ' ');
565:C      5      END;
566:C      4      END;
567:C      4      KPUTFIRST:
568:C      4      BEGIN { CALL ONLY IF SIZE <= MAXSIZE }
569:C      4      IF ECHO THEN
570:C      5      BEGIN
571:C      5      POSITION := 0;
572:C      5      CALL (CRTLLHOOK, CLLSHIFTR, POSITION, ' ');
573:C      5      CALL (CRTLLHOOK, CLLPUT, POSITION, C);
574:C      5      END;
575:C      4      BACKUP (OUTP); BUFFER^[OUTP] := C; SIZE := SIZE+1;
576:C      4      END;
577:C      4      END; { CASE }
578:C      2      END; { KEYBUFOPS }
579:S
580:D      1      PROCEDURE SETSTATUS (N: INTEGER; C: CHAR);
581:D      -1      VAR C1: CHAR;
582:C      2      BEGIN STATUSLINE[N] := C; C1 := C; CALL (CRTLLHOOK, PUTSTATUS, N, C1);
583:C      2      END;
584:D      1      FUNCTION RUNLIGHT: CHAR;
585:C      2      BEGIN RUNLIGHT := STATUSLINE[7];
586:C      2      END;
587:D      1      PROCEDURE SETRUNLIGHT (C: CHAR);
588:C      2      BEGIN SETSTATUS (7, C);
589:C      2      END;
590:S
591:D      -308  1  (* HPHIL *****
592:D      -308  1  DUMMY AODED 4/6/84 SFB)
593:D      1      PROCEDURE DUMMYHPHILCMD (OP: HPHILOP);
594:C      2      BEGIN END;
595:S
596:D      -308  1  (* INITIALIZE ALL HOOKS *****
597:D      1      PROCEDURE SYSDEV_INIT;
598:D      -2      VAR I: SHORTINT;
599:C      2      BEGIN { SYSDEV_INIT }
600:C      2      MASKOPSHOOK := DUMMYOUT2;

```



```

601:C      2      BEEPERHOOK := DUMMYOUT2;
602:S
603:C      2      RPPREQHOOK := DUMMYREQ1;
604:C      2      RPPISRHOOK := DUMMYKBD;
605:S
606:C      2      BATREADHOOK := DUMMYBATREAD;
607:C      2      BATCMDHOOK := DUMMYBATCMD;
608:S
609:C      2      CLOCKREQHOOK := DUMMYCLOCKREQ;
610:C      2      CLOCKIOHOOK := DUMMYCLOCKIO;
611:C      2      TIMERIOHOOK := DUMMYTIMERIO;
612:C      2      TIMERISRHOOK := DUMMYKBD;
613:S
614:C      2      KBDREQHOOK := DUMMYREQ1;
615:C      2      KBDISRHOOK := DUMMYKBD;
616:C      2      KBDTYPE := NOKBD;
617:C      2      KBDLANG := NO_KBD;
618:C      2      KBDIOHOOK := DUMMYTM;
619:C      2      KBDPOLHOOK := DUMMYB00LPROC;
620:C      2      MENUSTATE := M_NONE;
621:S
622:C      2      TOGGLEALPHAHOOK := DUMMYPROC;
623:C      2      TOGGLEGRAPHICSHOOK := DUMMYPROC;
624:C      2      DUMPALPHAHOOK := DUMMYPROC;
625:C      2      DUMPGRAPHICSHOOK := DUMMYPROC;
626:C      2      UPDATECURSORHOOK := DUMMYPROC;
627:C      2      CRTINITHOOK := DUMMYPROC;
628:C      2      CRTIOHOOK := DUMMYTM;
629:C      2      CRTLLHOOK := DUMMYCRTLL;
630:C      2      DBCRTHOOK := DUMMYDBCRT;
631:C      2      CURRENTCRT := NOCRT;
632:C      2      NEW(SYSCOM);
633:S
634:C      2      NEW(KEYBUFFER);
635:C      2      WITH KEYBUFFER^ DO
636:C      3      BEGIN
637:C      3      NEW(BUFFER);
638:C      3      ECHO := TRUE;
639:C      3      MAXSIZE := MAXBUFSIZE;
640:C      3      INP:=0; OUTP:=0; SIZE:=0;
641:C      3      NON_CHAR := ' ';
642:C      3      END;
643:S
644:C      2      FOR I:=0 TO 7 DO STATUSLINE[I]:=' ';
645:C      2      KBDWAITHOOK := DUMMYPROC;
646:C      2      KBDRELEASEHOOK := DUMMYPROC;
647:S
648:C      2      FOR I:=0 TO 1 DO LANGTABLE[I]:=NIL;
649:C      2      LANGINDEX:=0;
650:C      2      KBDYSYMODE:=TRUE; KBDALTLOCK:=FALSE; KBDCAPSLOCK:=FALSE;
651:C      2      KBDTRANSHOOK := DUMMYKBD; TRANSMODE :=KPASSTHRU;
652:S
653:C      2      HPHILCMDHOOK := DUMMYHPHILCMD; {4/6/84 SFB}
654:C      2      LOOPCONTROL := NIL; {4/9/84 SFB}
655:C      2      LOOPDRIVERLIST := NIL; {4/9/84 SFB}
656:S
657:C      2      END; {SYSDEV_INIT}
658:S
659:C      1      END; {MODULE}
660:S

```

```

661:D      1      IMPORT SYSDEVS;
662:C      1      BEGIN
663:C      1      SYSDEV_INIT;
664:C      1      {MARKUSER}
665:C      1      END. {PROGRAM}

```

No errors. No warnings.

***** Nonstandard language features enabled *****

TAIL

Description

TAIL invokes the loader to execute TABLE and STARTUP.

Usage

TAIL is placed at the end of INITLIB, since it terminates the boot-up process.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:S      (*
2:S
3:S      (c) Copyright Hewlett-Packard Company, 1984.
4:S      All rights are reserved. Copying or other
5:S      reproduction of this program except for archival
6:S      purposes is prohibited without the prior
7:S      written consent of Hewlett-Packard Company.
8:S
9:S
10:S     RESTRICTED RIGHTS LEGEND
11:S
12:S     Use, duplication, or disclosure by the Government
13:S     is subject to restrictions as set forth in
14:S     paragraph (b) (3) (B) of the Rights in Technical
15:S     Data and Computer Software clause in
16:S     DAR 7-104.9(a).
17:S
18:S     HEWLETT-PACKARD COMPANY
19:D     0 Fort Collins, Colorado      *)
20:S
21:S
22:D     0 $modcal$
23:D     0 $heap_dispose off$
24:D     0 $iocheck off$
25:D     0 $range off$ $ovflcheck off$
26:D     0 $debug off$
27:D     0 $stackcheck off$
28:D     0 $search 'INITLOAD', 'ASM', 'INIT', 'SYSDEVS'$
29:S
30:D     0 program tail(output);
31:S
32:D     1 import sysglobals, loader, ldr, misc, asm, bootdammodule, sysdevs;
33:S
34:D     -66 1 var start, table, systemname: string[20];
35:S
36:C     1 begin
37:S
38:C     1 ci_switch;           (move onto supervisor stack)
39:S
40:C     1 markuser;           (retain memory allocation from this point)
41:S
42:S     (*****
43:C     1 INTRODUCTORY MESSAGE )
44:S
45:C     1   writeln('Copyright 1984 Hewlett-Packard Company. ');
46:S
47:S     (*****
48:C     1 COMPUTE NAMES OF STANDARD FILES)
49:C     1
50:C     1   start := 'STARTUP'; table := 'TABLE';
51:C     1   setstrlen(systemname, 0); strmove(10, sysname, 1, systemname, 1);
52:C     1   systemname := strtrim(systemname);
53:C     1   if systemname<>'SYSTEM_P' then
54:C     1     begin
55:C     2     setstrlen(start, 5);
***WARNING: (line 56) STRPOS does not conform to HP standard, see $SWITCH_STRPOS$
56:C     2     if strpos('SYS', systemname)=1 then
57:C     3     begin
58:C     3     strdelete(systemname, 1, 3);
***WARNING: (line 58) STRPOS does not conform to HP standard, see $SWITCH_STRPOS$

```

```

59:C     3   if strpos('TEM_', systemname)=1 then strdelete(systemname, 1, 4);
60:C     3   end;
61:C     2   start := start + systemname;
62:C     2   table := table + systemname;
63:C     2   end;
64:S
65:S     (*****
66:C     1 LOAD MAIN PROGRAM )
67:S
68:C     1   try load(nodestr+start, true);
69:C     2   recover if escapecode<>-1 then escape(escapecode);
70:C     1   markuser;
71:S
72:S     (*****
73:C     1 ALLOW RECONFIGURATION OF UNIT TABLE)
74:S
75:C     1   try load(nodestr+table, false);   go;   (load new unit table)
76:C     2   recover if escapecode<>-1 then escape(escapecode);
77:C     1   releaseuser;
78:S
79:S     (*****
80:C     1 END OF SYSTEM INITIALIZATION)
81:C     1
82:C     1
83:S
84:C     1   try
85:C     2   loadrom(start);
86:C     2   if entrypoint = nil then load('*'+start, false);
87:C     2   CALL(MASKOPSHOOK,KBDMASK*RESETMASK,0); (enable the keyboard)
88:C     2   go;   (run the program)
89:C     2   recover printerror(escapecode, ioreult);
90:C     1
91:C     1   writeln(output, 'SYSTEM FINISHED');
92:C     1   CALL(MASKOPSHOOK,KBDMASK*RESETMASK,0); (enable the keyboard)
93:C     1   while true do try while true do; recover;   (TRY TO TRAP STOP KEY)
94:S
95:C     1 end.
96:S

```

No errors. 2 warnings.

***** Nonstandard language features enabled *****

TAPEBKUP

Description

TAPEBKUP is the Command Set '80 disc/tape backup utility program.

Usage

TAPEBKUP is used for complete CS80 media backup and restoration, certification of CS80 tapes and complete media verification of both discs and tapes.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).


```

1:D 0 $modcal, switch_strpos, debug off, range off, ovflcheck off$
2:D 0
3:D 0 $search 'IOL18:KERNEL', 'INIT:DRVASH', 'INIT:DISCHPIB',
4:D 0 'INIT:CS80'$
5:S
6:S
7:S
8:D 0 program tapebkup(keyboard,input,output); (CS80 streaming backup tape utility)
9:S
10:S
11:D 1 module CS80tbr; (tape backup routines)
12:S
13:D 1 import
14:D 1 sysglobals, bkgnd, discHPIB, CS80;
15:S
16:D 1 export
17:D 1
18:S
19:S
20:S { NOTE: the following functions each perform a COMPLETE transaction. They:
21:S . issue a (device or transparent) command (Command message)
22:S . transfer data if applicable (Execution message)
23:S . return the resulting QSTAT (Reporting message)
24:D 1 }
25:D 1 function set_specified_unitvol (uep: uep_type; unit, vol: unsn4): unsn8;
26:D 1 function initialize_media (uep: uep_type; options, interleave: unsn8): unsn8;
27:D 1 function set_address_and_return_mode (uep: uep_type; address: integer): unsn8;
28:D 1 function write_file_mark (uep: uep_type): unsn8;
29:D 1 function copy_start_address (uep: uep_type; var csa_bytes: sva_type): unsn8;
30:D 1 function unload (uep: uep_type): unsn8;
31:D 1 function locate_and_verify (uep: uep_type; length: integer): unsn8;
32:D 1 function volume_copy_data (source_uep: uep_type;
33:D 2 source_bk_addr: integer;
34:D 2 destination_uep: uep_type;
35:D 2 destination_bk_addr: integer): unsn8;
36:S
37:D 1 implement (CS80tbr)
38:S
39:S
40:D 1 type
41:D 1 setunitvol_type = (SET_UNIT/SET_VOLUME command pair)
42:D 1 packed record
43:D 1 setunit: CMD_type;
44:D 1 setvol: CMD_type;
45:D 1 end;
46:S
47:S
48:D 1 function suv_CMD_pair(uep: uep_type): setunitvol_type;
49:C 2 begin {suv_CMD_pair}
50:C 2 suv_CMD_pair.setunit := CMD_type(signed16(CMDset_unit_0)+uep^.du);
51:C 2 suv_CMD_pair.setvol := CMD_type(signed16(CMDset_vol_0)+uep^.dv);
52:C 2 end; {suv_CMD_pair}

```

```

53:D 1 $page$
54:S
55:C 1 function set_specified_unitvol(uep: uep_type; unit, vol: unsn4): unsn8;
56:C 2 {
57:C 2 . issue the following command sequence:
58:C 2 . SET_UNIT
59:C 2 . SET_VOLUME
60:C 2 return the QSTAT byte
61:D 2 }
62:D 2 var
63:D 2 suv: setunitvol_type;
64:C -2 2 begin {set_specified_unitvol}
65:C 2 suv.setunit := CMD_type(signed16(CMDset_unit_0)+unit);
66:C 2 suv.setvol := CMD_type(signed16(CMDset_vol_0)+vol);
67:C 2 HPIBshort_msge_out(uep, command_sec, addr(suv), sizeof(suv));
68:C 2 HPIBwait_for_ppol(uep);
69:C 2 set_specified_unitvol := qstat(uep);
70:C 2 end; {set_specified_unitvol}
71:S
72:S
73:D 1 function initialize_media(uep: uep_type; options, interleave: unsn8): unsn8;
74:S
75:C 2 {
76:C 2 . issue the following command sequence:
77:C 2 . SET_UNIT
78:C 2 . SET_VOLUME
79:C 2 . INITIALIZE_MEDIA
80:C 2 return the QSTAT byte
81:D 2 }
82:D 2 var
83:D 2 im: (the S bytes in the command message)
84:D 2 packed record
85:D 2 setunitvol: setunitvol_type;
86:D 2 initmedia: CMD_type;
87:D 2 options: unsn8;
88:D -6 2 interleave: unsn8;
89:C 2 end;
90:C 2 begin {initialize_media}
91:C 2 im.setunitvol := suv_CMD_pair(uep);
92:C 2 im.initmedia := CMDinit_media;
93:C 2 im.options := options;
94:C 2 im.interleave := interleave;
95:C 2 HPIBshort_msge_out(uep, command_sec, addr(im), sizeof(im));
96:C 2 HPIBwait_for_ppol(uep);
97:C 2 initialize_media := qstat(uep);
98:C 2 end; {initialize_media}

```



```

98:D      1 $page$
99:S
100:D      1 function set_address_and_return_mode(uep: uep_type; address: integer): unsgn8;
101:S      {
102:S          issue the following command sequence:
103:S              . SET_UNIT
104:S              . SET_VOLUME
105:S              . SET_ADDRESS_RETURN_MODE
106:S              . SET_ADDRESS
107:S          return the QSTAT byte
108:D      }
109:D      2
110:D      var
111:D          sarm: (the 12 bytes in the command message)
112:D          packed record
113:D              setunitvol: setunitvol_type;
114:D              setretadd: CMD_type;
115:D              retaddmode: unsgn8;
116:D              nop: CMD_type;
117:D              setadd: CMD_type;
118:D              sva: sva_type;
119:C      -12
120:C      begin (set_address_and_return_mode)
121:C          sarm.setunitvol := suv_CMD_pair(uep);
122:C          sarm.setretadd := CMDset_retadd_mode;
123:C          sarm.retaddmode := 0; {single vector;}
124:C          sarm.nop := CMDno_op;
125:C          sarm.setadd := CMDset_address_1V;
126:C          sarm.sva.utb := 0;
127:C          sarm.sva.lfb := address;
128:C          HPI8short_msge_out(uep, command_sec, addr(sarm), sizeof(sarm));
129:C          HPI8wait_for_ppol(uep);
130:C          set_address_and_return_mode := qstat(uep);
131:C      end; {set_address_and_return_mode}
132:S
133:D      1 function write_file_mark(uep: uep_type): unsgn8;
134:S      {
135:S          issue the following command sequence:
136:S              . SET_UNIT
137:S              . SET_VOLUME
138:S              . WRITE_FILE MARK
139:S          return the QSTAT byte
140:D      }
141:D      2
142:D      var
143:D          wfm: (the 3 bytes in the command message)
144:D          packed record
145:D              setunitvol: setunitvol_type;
146:D              writefilemark: CMD_type;
147:C      -4
148:C      begin (write_file_mark)
149:C          wfm.setunitvol := suv_CMD_pair(uep);
150:C          wfm.writefilemark := CMDwrite_file_mark;
151:C          HPI8short_msge_out(uep, command_sec, addr(wfm), sizeof(wfm));
152:C          HPI8wait_for_ppol(uep);
153:C          write_file_mark := qstat(uep);
154:C      end; {write_file_mark}

```

```

154:D      1 $page$
155:S
156:D      1 function unload(uep: uep_type): unsgn8;
157:S      {
158:S          issue the following command sequence:
159:S              . SET_UNIT
160:S              . SET_VOLUME
161:S              . UNLOAD
162:S          return the QSTAT byte
163:D      }
164:D      2
165:D      var
166:D          unl: (the 3 bytes in the command message)
167:D          packed record
168:D              setunitvol: setunitvol_type;
169:D              unloadcmd: CMD_type;
170:C      -4
171:C      begin (unload)
172:C          unl.setunitvol := suv_CMD_pair(uep);
173:C          unl.unloadcmd := CMDunload;
174:C          HPI8short_msge_out(uep, command_sec, addr(unl), sizeof(unl));
175:C          HPI8wait_for_ppol(uep);
176:C          unload := qstat(uep);
177:C      end; {unload}
178:S
179:D      1 function locate_and_verify(uep: uep_type; length: integer): unsgn8;
180:S      {
181:S          issue the following command sequence:
182:S              . SET_UNIT
183:S              . SET_VOLUME
184:S              . SET_LENGTH
185:S              . LOCATE AND VERIFY
186:S          return the QSTAT byte
187:D      }
188:D      2
189:D      var
190:D          lv: (the 9 bytes in the command message)
191:D          packed record
192:D              setunitvol: setunitvol_type;
193:D              nop: CMD_type;
194:D              setlen: CMD_type;
195:D              len: integer;
196:D              locateandverify: CMD_type;
197:C      -10
198:C      begin (locate_and_verify)
199:C          lv.setunitvol := suv_CMD_pair(uep);
200:C          lv.nop := CMDno_op;
201:C          lv.setlen := CMDset_length;
202:C          lv.len := length;
203:C          lv.locateandverify := CMDlocate_and_ver;
204:C          HPI8short_msge_out(uep, command_sec, addr(lv), sizeof(lv));
205:C          HPI8wait_for_ppol(uep);
206:C          locate_and_verify := qstat(uep);
207:C      end; {locate_and_verify}

```



```

207:D      1 $page$
208:S
209:D      1 function volume_copy_data(source_uep: uep_type; source_blk_addr: integer;
210:D      2 destination_uep: uep_type; destination_blk_addr: integer): unsgn8;
211:S      {
212:S          issue the following command sequence:
213:S          . SET_UNIT 15
214:S          . SET_LENGTH
215:S          . COPY_DATA
216:S          return the QSTAT byte
217:D      2 }
218:D      2 var
219:D      2 cd: (the 24 bytes in the command message)
220:D      2 packed record
221:D      2     setunit15: CMD_type;
222:D      2     setlen: CMD_type;
223:D      2     length: integer;
224:D      2     nop: CMD_type;
225:D      2     copydata: CMD_type;
226:D      2     s_volunit: evu_type;
227:D      2     s_setadd: CMD_type;
228:D      2     s_address: sva_type;
229:D      2     d_volunit: evu_type;
230:D      2     d_setadd: CMD_type;
231:D      2     d_address: sva_type;
232:D      2 end;
233:C      -24 begin (volume_copy_data)
234:C      2 cd.setunit15 := CMDset_unit_15;
235:C      2 cd.setlen := CMDset_length;
236:C      2 cd.length := -1; (full volume)
237:C      2 cd.nop := CMDno_op;
238:C      2 cd.copydata := CMDcopy_data;
239:C      2 cd.s_volunit.vvvv := source_uep^.dv;
240:C      2 cd.s_volunit.uuuu := source_uep^.du;
241:C      2 cd.s_setadd := CMDset_address_1v;
242:C      2 cd.s_address.utb := 0;
243:C      2 cd.s_address.lfb := source_blk_addr;
244:C      2 cd.d_volunit.vvvv := destination_uep^.dv;
245:C      2 cd.d_volunit.uuuu := destination_uep^.du;
246:C      2 cd.d_setadd := CMDset_address_1v;
247:C      2 cd.d_address.utb := 0;
248:C      2 cd.d_address.lfb := destination_blk_addr;
249:C      2 HP18short_msg_out(source_uep, command_sec, addr(cd), sizeof(cd));
250:C      2 HP18wait_for_ppol(source_uep);
251:C      2 volume_copy_data := qstat(source_uep);
252:C      2 end; (volume_copy_data)

```

```

253:D      1 $page$
254:S
255:D      1 function copy_start_address(uep: uep_type; var csa_bytes: sva_type): unsgn8;
256:S      {
257:S          issue the following command sequence:
258:S          . SET_UNIT
259:S          . SET_VOLUME
260:S          . READ_DRIVE TABLES: COPY START ADDRESS (7914 ONLY!!!)
261:S          return the QSTAT byte
262:D      2 }
263:D      2 var
264:D      2 csa: (the 5 bytes in the command message)
265:D      2 packed record
266:D      2     setunitvol: setunitvol_type;
267:D      2     initutil: CMD_type;
268:D      2     readdrivetables: unsgn8;
269:D      2     tablenumber: unsgn8;
270:D      2 end;
271:D      -6 premature_eoi: boolean;
272:D      -7 qs: unsgn8;
273:C      -10 begin (copy_start_address)
274:C      2 csa.setunitvol := suv_CMD_pair(uep);
275:C      2 csa.initutil := CHDInit_util_SEH; (send execution message)
276:C      2 csa.readdrivetables := 196;
277:C      2 csa.tablenumber := 12;
278:C      2 HP18short_msg_out(uep, command_sec, addr(csa), sizeof(csa));
279:C      2 HP18wait_for_ppol(uep);
280:C      2 try
281:C      3 HP18short_msg_in(uep, execution_sec, addr(csa_bytes), sizeof(csa_bytes));
282:C      3 premature_eoi := false;
283:C      3 recover
284:C      4 with btp_type(uep^.dvrtemp)^ do (confirm the "premature" eoi)
285:C      4 begin
286:C      5 if (escapecode<-10) or (iores<zbaddhardware) then
287:C      5 escape(escapecode);
288:C      4 premature_eoi := true;
289:C      4 iores := inoerror;
290:C      4 end; (with)
291:C      3 HP18wait_for_ppol(uep);
292:C      3 qs := qstat(uep);
293:C      3 if premature_eoi and (qs=0) then
294:C      3 ioresc_bkgnd(uep, zcatchall);
295:C      2 copy_start_address := qs;
296:C      2 end; (copy_start_address)
297:S
298:S
299:C      1 end; (CS80tbr)

```



```

300:D      1 $page$
301:S
302:D      1 module C580tdvdr; {tape backup driver}
303:S
304:D      1 import
305:D      1   sysglobals, misc, bkgnd, discHPI8, tapebuf, C580, C580dsr, C580tbr;
306:S
307:D      1 export
308:D      1   type
309:D      1   uep_proc_type = procedure(uep: uep_type);
310:S
311:D      1   var {"var parameter" set by Qdescribe}
312:D      -38 1   describe_bytes: describe_type;
313:D      -38 1   procedure Qdescribe(uep: uep_type);
314:S
315:D      -38 1   function Qdevicename(uep: uep_type; var saved_ioresult: integer): string80;
316:S
317:D      -38 1   var {"parameters" for Qverify}
318:D      -42 1   verify_length: integer;
319:D      -46 1   verify_block_size: integer;
320:D      -46 1   procedure Qverify(uep: uep_type);
321:S
322:D      -46 1   procedure Qtapesetup(uep: uep_type);
323:D      -46 1   procedure Qconfigure(uep: uep_type);
324:D      -46 1   procedure Qwritefilemark(uep: uep_type);
325:D      -46 1   procedure Qunload(uep: uep_type);
326:S
327:D      -46 1   var {"parameter" for Qcertify}
328:D      -48 1   initialize_options_byte: unsng8;
329:D      -48 1   procedure Qcertify(uep: uep_type);
330:S
331:D      -48 1   var {"parameters" for Qcontrollercopy}
332:D      -52 1   destination_uep: uep_type;
333:D      -56 1   source_block_address: integer;
334:D      -60 1   destination_block_address: integer;
335:D      -60 1   procedure Qcontrollercopy(source_uep: uep_type);
336:S
337:D      -60 1   var {"var parameter" for Qcopy_start_address}
338:D      -66 1   csa_bytes: sva_type;
339:D      -66 1   procedure Qcopy_start_address(uep: uep_type);
340:S
341:S
342:D      -66 1 implement {C580tdvdr}

```

```

343:D      -66 1 $page$
344:S
345:D      1 procedure with_bkgnd_do(uep_proc: uep_proc_type; uep: uep_type);
346:C      2   begin {with_bkgnd_do}
347:C      2   try
348:C      3   ioresult := ord(ioerror);
349:C      3   allocate_bkgnd_info(uep);
350:C      3   if HPI8amigo_identify(uep) div 256 <> 2 then
351:C      4   ioresc_bkgnd(uep, znodvice);
352:C      3   call(uep_proc, uep);
353:C      3   deallocate_bkgnd_info(uep);
354:C      3   recover
355:C      3   begin
356:C      3   abort_bkgnd_process(uep);
357:C      3   ioresult := uep^.dvrtmp;
358:C      3   uep^.dvrtmp := ord(ioerror); {report the error only once}
359:C      3   escape[10];
360:C      3   end; {recover}
361:C      2   end; {with_bkgnd_do}
362:S
363:S
364:D      1 procedure Qtapesetup(uep: uep_type);
365:S
366:D      2   procedure tapesetup(uep: uep_type);
367:D      3   var
368:D      -1 3   retry_required: boolean;
369:C      3   begin {tapesetup}
370:C      3   repeat
371:C      4   retry_required := false;
372:C      4   if set_unitvol(uep) <> 0 then
373:C      5   handle_bad_status(uep, true, retry_required);
374:C      4   until not retry_required;
375:C      3   repeat {enable auto skip sparing}
376:C      4   retry_required := false;
377:C      4   if set_options(uep, (options byte) 6) <> 0 then
378:C      5   handle_bad_status(uep, true, retry_required);
379:C      4   until not retry_required;
380:C      3   end; {tapesetup}
381:S
382:C      2   begin {Qtapesetup}
383:C      2   with_bkgnd_do(tapesetup, uep);
384:C      2   end; {Qtapesetup}
385:S
386:S
387:D      1 procedure Qconfigure(uep: uep_type);
388:C      2   begin {Qconfigure}
389:C      2   with_bkgnd_do(configure, uep);
390:C      2   end; {Qconfigure}

```



```

391:D -66 1 $page$
392:S
393:D 1 procedure Qwritefilemark(uep: uep_type);
394:S
395:D 2 procedure writefilemark(uep: uep_type);
396:D 3 var
397:D -1 3 retry_required: boolean;
398:C 3 begin (writefilemark)
399:C 3 repeat
400:C 4 retry_required := false;
401:C 4 if write_file_mark(uep) <> 0 then
402:C 5 handle_bad_status(uep, true, retry_required);
403:C 4 until not retry_required;
404:C 3 end; (writefilemark)
405:S
406:C 2 begin (Qwritefilemark)
407:C 2 with bkgnd_do(writefilemark, uep);
408:C 2 end; (Qwritefilemark)
409:S
410:S
411:D 1 procedure Qcertify(uep: uep_type);
412:S
413:D 2 procedure certify(uep: uep_type);
414:D 3 var
415:D -1 3 retry_required: boolean;
416:C 3 begin (certify)
417:C 3 repeat (initialize the medium)
418:C 4 retry_required := false;
419:C 4 if initialize_media(uep, initialize_options_byte, (interleave_factor) 0) <> 0 then
420:C 5 handle_bad_status(uep, true, retry_required);
421:C 4 until not retry_required;
422:C 3 end; (certify)
423:S
424:C 2 begin (Qcertify)
425:C 2 with bkgnd_do(certify, uep);
426:C 2 end; (Qcertify)
427:S
428:S
429:D 1 procedure Qdescribe(uep: uep_type);
430:S
431:D 2 procedure do_describe(uep: uep_type);
432:D 3 var
433:D -1 3 retry_required: boolean;
434:C 3 begin (do_describe)
435:C 3 repeat
436:C 4 retry_required := false;
437:C 4 if set_unitvol(uep) <> 0 then
438:C 5 handle_bad_status(uep, true, retry_required);
439:C 4 until not retry_required;
440:C 3 repeat
441:C 4 retry_required := false;
442:C 4 if describe(uep, describe_bytes) <> 0 then
443:C 5 handle_bad_status(uep, true, retry_required);
444:C 4 until not retry_required;
445:C 3 end; (do_describe)
446:S
447:C 2 begin (Qdescribe)
448:C 2 with bkgnd_do(do_describe, uep);
449:C 2 end; (Qdescribe)

```

```

450:D -66 1 $page$
451:S
452:D 1 function Qdevicename(uep: uep_type; var saved_ioresult: integer): string80;
453:S
454:D 2 var
455:D -82 2 devicename: string80;
456:D -86 2 index: integer;
457:D -86 2 bcd_product_number: (within the describe bytes)
458:D -86 2 packed record case integer of
459:D -86 2 0: (dn: usgn24);
460:D -86 2 1: (bcd: packed array[1..6] of usgn4);
461:D -80 2 end;
462:D -94 2 product_number: integer;
463:D -94 2 fixed_vol_byte:
464:D -94 2 packed record case integer of
465:D -84 2 0: (byte: usgn8);
466:D -94 2 1: (bit: packed array[0..7] of boolean);
467:D -96 2 end;
468:S
469:C 2 begin (Qdevicename)
470:S
471:C 2 try
472:C 3 Qdescribe(uep);
473:C 3 recover
474:C 3 if escapecode <> -10 then
475:C 4 escape(escapecode);
476:S
477:C 2 if saved_ioresult = ord(inoerror) then (report any error with describe)
478:C 3 saved_ioresult := ioresult; (can be inoerror)
479:S
480:C 2 if ioresult = ord(inoerror) then
481:C 3 with describe_bytes do
482:C 4 begin
483:C 4 bcd_product_number.dn := dn;
484:C 4 product_number := 0;
485:C 4 for index := 1 to 5 do
486:C 5 product_number := product_number*10 + bcd_product_number.bcd[index];
487:C 4 devicename := 'HP';
488:C 4 strwrite(devicename, 3, index, product_number:1);
489:C 4 case dn of
490:C 5 0: devicename := devicename + ' fixed disc';
491:C 5 1: begin
492:C 6 fixed_vol_byte.byte := describe_bytes.fvb;
493:C 6 if fixed_vol_byte.bit[7-uep^.dv]
494:C 6 then devicename := devicename + ' fixed disc'
495:C 6 else devicename := devicename + ' removeable disc';
496:C 6 end;
497:C 5 2: devicename := devicename + ' tape';
498:C 5 otherwise (can't further describe the generic device type);
499:C 5 end; (case)
500:C 4 end (with)
501:C 4 else
502:C 3 devicename := '<inaccessible C580 device>';
503:S
504:S
505:C 2 Qdevicename := devicename;
506:C 2 end; (Qdevicename)

```



```
507:D -66 1 $page$
508:S
509:D 1 procedure Qverify(uep: uep_type);
510:S
511:D 2 var
512:D -4 2 working_verify_length: integer;
513:D -5 2 bad_blocks_encountered: boolean;
514:S
515:D 2 procedure handle_verify_status(var retry_required: boolean);
516:S
517:D 3 var
518:D -2 3 iorval_to_report: iorsltwd; {to hold the first reportable error}
519:D -4 3 working_iorval: iorsltwd; {cleared each time status is read}
520:D -24 3 status_bytes: status_type;
521:D -28 3 eb_scan, parameter_field_owner: errorbit_type;
522:D -29 3 reconfiguration_needed: boolean;
523:S
524:S 3 begin {handle_verify_status}
525:S
526:C 3 iorval_to_report := inoerror;
527:S
528:C 3 repeat
529:S
530:C 4 if status(uep, status_bytes) <> 0 then
531:C 5 ioresc_bkgnd(uep, zbadhardware);
532:S
533:C 4 working_iorval := inoerror;
534:C 4 parameter_field_owner := channel_parity_error; {doesn't really own it!}
535:C 4 reconfiguration_needed := false;
536:S
537:C 4 for eb_scan := eb63 downto eb0 do
538:C 5 if status_bytes.errorbits[eb_scan] then
539:C 6 begin
540:S
541:C 6 if eb_scan in errorbits_owning_parmfield then
542:C 7 parameter_field_owner := eb_scan;
543:S
544:C 6 case eb_scan of
545:S
546:C 7 (specific fatal errors)
547:C 7 channel_parity_error,
548:C 7 controller_fault,
549:C 7 unit_fault,
550:C 7 diagnostic_result:
551:C 7 working_iorval := zbadhardware;
552:C 7 illegal_opcode,
553:C 7 parameter_bounds,
554:C 7 illegal_parameters:
555:C 7 working_iorval := zbadmode; {some cmds optional in 55/80}
556:C 7 module_addressing:
557:C 7 working_iorval := znodevice;
558:C 7 address_bounds,
559:C 7 end_of_volume:
560:C 7 working_iorval := znosuchblk;
561:C 7 uninitialized_media:
562:C 7 if status_bytes.errorbits[power_fail]
563:C 8 then {probably an uncertified tape; allow access anyway}
564:C 8 else working_iorval := zuninitialized;
565:C 7 no_spare_available:
566:C 7 working_iorval := zinitfail;
```

```
567:C 7 not_ready:
568:C 7 working_iorval := znoready;
569:C 7 write_protect:
570:C 7 working_iorval := zprotected;
571:C 7 no_data_found:
572:C 7 end_of_file:
573:C 7 working_iorval := znoblock;
574:C 7 unrecoverable_data_overflow,
575:C 7 unrecoverable_data:
576:C 7 working_iorval := zbadblock;
577:S
578:C 7 {power fail}
579:C 7 power_fail:
580:C 7 begin
581:C 7 uep^.umediavalid := false;
582:C 7 if uep=tapebuf_uep then
583:C 8 tapebuf_state := undefined;
584:C 7 if uep^.ureportchange then
585:C 8 working_iorval := zmediumchanged;
586:C 7 reconfiguration_needed := true;
587:C 7 retry_required := true;
588:C 7 end;
589:S
590:C 7 {retryable errors}
591:C 7 operator_release_required,
592:C 7 diagnostic_release_required,
593:C 7 internal_maintenance_required,
594:C 7 retransmit:
595:C 7 retry_required := true;
596:S
597:C 7 {errors indicating release requested}
598:C 7 operator_request,
599:C 7 diagnostic_request,
600:C 7 internal_maintenance_request:
601:C 7 {do nothing here; release below if parameter field owned};
602:S
603:C 7 {errors indicating reconfiguration needed}
604:C 7 media_wear, {supposed to be masked out}
605:C 7 latency_induced, {supposed to be masked out}
606:C 7 eb53, {supposed to be masked out}
607:C 7 eb54, {supposed to be masked out}
608:C 7 auto_sparing_invoked, {supposed to be masked out}
609:C 7 eb56, {supposed to be masked out}
610:C 7 recoverable_data_overflow, {supposed to be masked out}
611:C 7 marginal_data, {supposed to be masked out}
612:C 7 recoverable_data, {supposed to be masked out}
613:C 7 eb80, {supposed to be masked out}
614:C 7 maintenance_track_overflow, {supposed to be masked out}
615:C 7 eb62, {supposed to be masked out}
616:C 7 eb63, {supposed to be masked out}
617:C 7 reconfiguration_needed := true;
618:S
619:C 7 {errors not covered by the above cases}
620:C 7 otherwise
621:C 8 {specifically including:
622:C 8 message_sequence,
623:C 8 message_length,
624:C 8 cross_unit,
625:C 8 illegal_parallel_operation }
626:C 7 working_iorval := zcathall;
```



```

627:S
628:C      7      end; {case}
629:S
630:C      6      end; {if}
631:S
632:C      4      if iorval_to_report=inoerror then {none previously found; report this one}
633:C      5      iorval_to_report := working_iorval; {it can be inoerror also!}
634:S
635:C      4      if parameter_field_owner=unrecoverable_data then
636:C      5      begin
637:C      5      writeLn(' bad block at ', status_bytes.bba.lfb:1);
638:C      5      bad_blocks_encountered := true;
639:C      5      working_verify_length := verify_length-(status_bytes.bba.lfb:1)*verify_block_size;
640:C      5      if working_verify_length>0 then
641:C      6      retry_required := true;
642:C      5      end; {if}
643:S
644:C      4      if parameter_field_owner in errorbits_requesting_release then
645:C      5      if not (status_bytes.urr[1] in [0..15]) then
646:C      6      ioresc_bkgnd(uep, zcatchall)
647:C      6      else if release(uep, status_bytes.urr[1])<>0 then
648:C      7      {handle the bad qstat elsewhere; worry not, the device won't forget!};
649:S
650:C      4      if reconfiguration_needed then
651:C      5      configure(uep);
652:S
653:C      4      until set_unit(uep, status_bytes.current_vu.uuuu)=0; {restore original command unit}
654:S
655:C      3      if not (iorval_to_report in [inoerror, zbadblock]) then
656:C      4      ioresc_bkgnd(uep, iorval_to_report);
657:S
658:C      3      end; {handle_verify_status}

```

```

659:D      -5 2 $page$
660:S
661:D      2      procedure verify(uep: uep_type);
662:S
663:D      3      var
664:D      -1 3      retry_required: boolean;
665:S
666:C      3      begin {verify}
667:S
668:C      3      repeat {set address to zero and set .return addressing mode}
669:C      4      retry_required := false;
670:C      4      if set_address_and_return_mode(uep, {address} 0)<>0 then
671:C      5      handle_bad_status(uep, true, retry_required);
672:C      4      until not retry_required;
673:S
674:C      3      working_verify_length := verify_length;
675:C      3      bad_blocks_encountered := false;
676:S
677:C      3      repeat {verify the requested amount}
678:C      4      retry_required := false;
679:C      4      if locate_and_verify(uep, working_verify_length)<>0 then
680:C      5      handle_verify_status(retry_required);
681:C      4      until not retry_required;
682:S
683:C      3      if bad_blocks_encountered then
684:C      4      ioresc_bkgnd(uep, zbadblock);
685:S
686:C      3      end; {verify}
687:S
688:C      2      begin {Qverify}
689:C      2      with bkgnd_do(verify, uep);
690:C      2      end; {Qverify}

```



```

691:D -66 1 $pages
692:S
693:D 1 procedure Qcontrollercopy(source_uep: uep_type);
694:S
695:D 2 procedure handle_copydata_status(var retry_required: boolean);
696:S
697:D 3 var
698:D -2 3 iorval_to_report: iorsltwd; (to hold the first reportable error)
699:D -4 3 working_iorval: iorsltwd; (cleared each time status is read)
700:D -24 3 status_bytes: status_type;
701:D -28 3 eb_scan, parameter_field_owner: errorbit_type;
702:D -29 3 reconfiguration_needed: boolean;
703:D -30 3 uee_pending: boolean;
704:D -32 3 index: signed16;
705:S
706:D 3 procedure handle_uee_pending(uee_byte: signed8);
707:D 4 var
708:D 4 evu: {encoded volume/unit byte}
709:D 4 packed record case integer of
710:D 4 0: {sgn8: signed8};
711:D 4 1: {vu: evu_type};
712:D -2 4 end;
713:D -6 4 vol, unit: usgn4;
714:D -7 4 retry_required: boolean;
715:D -88 4 message: string80;
716:C 4 begin {handle_uee_pending}
717:C 4 evu.sgn8 := uee_byte;
718:C 4 vol := evu.vu.vvvv;
719:C 4 unit := evu.vu.uuuu;
720:C 4 try
721:C 5 if unit=15 then
722:C 6 repeat
723:C 7 retry_required := false;
724:C 7 if set_unit(source_uep, unit)<>0 then
725:C 8 handle_bad_status(source_uep, false, retry_required);
726:C 7 until not retry_required;
727:C 7 else
728:C 7 repeat
729:C 7 retry_required := false;
730:C 7 if set_specified_unitvol(source_uep, unit, vol)<>0 then
731:C 8 handle_bad_status(source_uep, false, retry_required);
732:C 7 until not retry_required;
733:C 5 recover
734:C 5 if escapecode<>-10 then escape(escapecode);
735:C 5 with btp_type(source_uep^ dvtemp)^ do
736:C 5 if iores<>inoerror then {we need to report it}
737:C 6 begin
738:C 6 if iorval_to_report=inoerror then {none previously found...}
739:C 7 iorval_to_report := iores; {report this one later in a fatal way!}
740:C 6 if (vol=source_uep^ dv) and (unit=source_uep^ du) then
741:C 7 writeln(' source unit error:');
742:C 6 else if (vol=destination_uep^ dv) and (unit=destination_uep^ du) then
743:C 7 writeln(' destination unit error:');
744:C 6 else
745:C 6 writeln(' unit: ', unit:1, ' volume: ', vol:1, ' error:');
746:C 6 getioerrmsg(message, ord(iores));
747:C 6 writeln(' ', message);
748:C 6 iores := inoerror; {now clear it out}
749:C 6 end; {if}
750:C 4 end; {handle_uee_pending}

```

```

751:S
752:C 3 begin {handle_copydata_status}
753:S
754:C 3 iorval_to_report := inoerror;
755:S
756:C 3 repeat
757:S
758:C 4 if status(source_uep, status_bytes)<>0 then
759:C 5 ioresc_bkgnd(source_uep, zbadhardware);
760:S
761:C 4 working_iorval := inoerror;
762:C 4 parameter_field_owner := channel_parity_error; {doesn't really own it!}
763:C 4 reconfiguration_needed := false;
764:S
765:C 4 for eb_scan := eb63 downto eb0 do
766:C 5 if status_bytes.errorbits[eb_scan] then
767:C 6 begin
768:S
769:C 6 if eb_scan in errorbits_owning_parmfield then
770:C 7 parameter_field_owner := eb_scan;
771:S
772:C 6 case eb_scan of
773:C 7 (the expected error)
774:C 7 cross_unit:
775:C 7 {do nothing here; handle below if parameter field owned};
776:C 7
777:C 7 (specific fatal errors)
778:C 7 channel_parity_error,
779:C 7 controller_fault,
780:C 7 unit_fault,
781:C 7 diagnostic_result:
782:C 7 working_iorval := zbadhardware;
783:C 7 illegal_opcode,
784:C 7 parameter_bounds,
785:C 7 illegal_parameter:
786:C 7 working_iorval := zbadmode; {some cmds optional in 55/80}
787:C 7 module_addressing:
788:C 7 working_iorval := znodvice;
789:C 7 address_bounds,
790:C 7 end_of_volume:
791:C 7 working_iorval := znosuchblk;
792:C 7 uninitialized_media:
793:C 7 if status_bytes.errorbits[power_fail]
794:C 8 then {probably an uncertified tape; allow access anyway}
795:C 8 else working_iorval := zuninitialized;
796:C 7 no_spare_available:
797:C 7 working_iorval := zinitfail;
798:C 7 not_ready:
799:C 7 working_iorval := znoready;
800:C 7 write_protect:
801:C 7 working_iorval := zprotected;
802:C 7 no_data_found,
803:C 7 end_of_file:
804:C 7 working_iorval := znoblock;
805:C 7 unrecoverable_data_overflow,
806:C 7 unrecoverable_data:
807:C 7 working_iorval := zbadblock;
808:C 7
809:S
810:C 7 (power fail)

```



```

811:C      7      power_fail:
812:C      7      begin
813:C      7          source_uep^.umediavalid := false;
814:C      7          if source_uep=tapebuf_uep then
815:C      8              tapebuf_state := undefined;
816:C      7          if source_uep^.ureportchange then
817:C      8              working_iorval := zmediumchanged;
818:C      7          reconfiguration_needed := true;
819:C      7          retry_required := true;
820:C      7      end;
821:S      7
822:C      7      (retryable errors)
823:C      7      operator_release_required,
824:C      7      diagnostic_release_required,
825:C      7      internal_maintenance_required,
826:C      7      retransmit:
827:C      7          retry_required := true;
828:S      7
829:C      7      (errors indicating release requested)
830:C      7      operator_request,
831:C      7      diagnostic_request,
832:C      7      internal_maintenance_request:
833:C      7          {do nothing here; release below if parameter field owned};
834:S      7
835:C      7      (errors indicating reconfiguration needed)
836:C      7      media_wear,                {supposed to be masked out}
837:C      7      latency_induced,           {supposed to be masked out}
838:C      7      eb53,                     {supposed to be masked out}
839:C      7      eb54,                     {supposed to be masked out}
840:C      7      auto_sparing_invoked,     {supposed to be masked out}
841:C      7      eb56,                     {supposed to be masked out}
842:C      7      recoverable_data_overflow, {supposed to be masked out}
843:C      7      marginal_data,            {supposed to be masked out}
844:C      7      recoverable_data,         {supposed to be masked out}
845:C      7      eb60,                     {supposed to be masked out}
846:C      7      maintenance_track_overflow, {supposed to be masked out}
847:C      7      eb63,                     {supposed to be masked out}
848:C      7      eb63,                     {supposed to be masked out}
849:C      7          reconfiguration_needed := true;
850:S      7
851:C      7      (errors not covered by the above cases)
852:C      7      otherwise
853:C      7          { specifically including:
854:C      7              message_sequence,
855:C      7              message_length,
856:C      7              illegal_parallel_operation }
857:C      7          working_iorval := zcatchall;
858:S      7
859:C      7      end; {case}
860:S      7
861:C      6      end; {if}
862:S      6
863:C      4      if iorval_to_report=inoerror then {none previously found; report this one}
864:C      5          iorval_to_report := working_iorval; {it can be inoerror also!}
865:S      5
866:C      4      index := 0;
867:C      4      if parameter_field_owner=cross_unit then
868:C      5          repeat
869:C      6              index := index+1;
870:C      6              if index<=6

```

```

871:C      7      then uee_pending := status_bytes.uee[index]>=0
872:C      7      else uee_pending := false;
873:C      6      if uee_pending then
874:C      7          handle_uee_pending(status_bytes.uee[index]);
875:C      6      until not uee_pending;
876:S      6
877:C      4      if parameter_field_owner in errorbits_requesting_release then
878:C      5          if not (status_bytes.urr[1] in [0..15]) then
879:C      6              ioresc_bkgnd[source_uep, zcatchall]
880:C      6          else if release(source_uep, status_bytes.urr[1])<>0 then
881:C      7              {handle the bad qstat elsewhere; worry not, the device won't forget!};
882:S      7
883:C      4      if reconfiguration_needed then
884:C      5          configure(source_uep);
885:S      5
886:C      4      until set_unit(source_uep, status_bytes.current_vu.uuuu)=0; {restore original command unit}
887:S      4
888:C      3      if iorval_to_report<>inoerror then
889:C      4          ioresc_bkgnd[source_uep, iorval_to_report];
890:S      4
891:C      3      end; {handle_copydata_status}
892:S      3
893:S      3
894:D      2      procedure controllercopy(uep: uep_type);
895:D      3      var
896:D      3          retry_required: boolean;
897:C      3      begin {controllercopy}
898:C      3          repeat {copy the entire volume}
899:C      4              retry_required := false;
900:C      4              if volume_copy_data(source_uep, source_block_address,
901:C      5                  destination_uep, destination_block_address)<>0 then
902:C      5                  handle_copydata_status(retry_required);
903:C      4              until not retry_required;
904:C      3          end; {controllercopy}
905:S      3
906:C      2      begin {Qcontrollercopy}
907:C      2          with_bkgnd_do(controllercopy, source_uep);
908:C      2      end; {Qcontrollercopy}

```



```

909:D -66 1 $page$
910:S
911:D 1 procedure Qunload(uep: uep_type);
912:S
913:D 2 procedure do_unload(uep: uep_type);
914:D 3 var
915:D -1 3 retry_required: boolean;
916:C 3 begin {do_unload}
917:C 3 repeat
918:C 4 retry_required := false;
919:C 4 if unload(uep) <> 0 then
920:C 5 handle_bad_status(uep, true, retry_required);
921:C 4 until not retry_required;
922:C 3 end; {do_unload}
923:S
924:C 2 begin {Qunload}
925:C 2 with_bkgnd_do(do_unload, uep);
926:C 2 end; {Qunload}
927:S
928:S
929:D 1 procedure Qcopy_start_address(uep: uep_type);
930:S
931:D 2 procedure do_copy_start_address(uep: uep_type);
932:D 3 var
933:D -1 3 retry_required: boolean;
934:C 3 begin {do_copy_start_address}
935:C 3 repeat
936:C 4 retry_required := false;
937:C 4 if copy_start_address(uep, csa_bytes) <> 0 then
938:C 5 handle_bad_status(uep, true, retry_required);
939:C 4 until not retry_required;
940:C 3 end; {do_copy_start_address}
941:S
942:C 2 begin {Qcopy_start_address}
943:C 2 with_bkgnd_do(do_copy_start_address, uep);
944:C 2 end; {Qcopy_start_address}
945:S
946:S
947:C 1 end; {C580tbdvr}

```

```

948:D 1 $page$
949:S
950:D 1 {program tapebkup;}
951:S
952:D 1 import
953:D 1 sysglobals, misc, sysdevs, fs, bkgnd, tapebuf, C580tbr, C580tbdvr;
954:S
955:D 1 var
956:D 1 keyboard: text;
957:S
958:D 1 type
959:D 1 proc_type = procedure;
960:S
961:D 1 const {for under_lock_do procedure calls}
962:D 1 verbosely = true;
963:D 1 silently = false; {if desired, set true to debug}
964:S
965:S
966:D 1 procedure upcchar(var ch : char);
967:C 2 begin
968:C 2 if (ch>='a') and (ch<='z') then ch := chr(ord(ch)-32);
969:C 2 end;
970:S
971:S
972:D -82 1 function yes(prompt: string80): boolean;
973:D -82 2 var
974:D -83 2 answer: char;
975:C 2 begin {yes}
976:C 2 writeln;
977:C 2 write(prompt, ' (Y/N) ');
978:C 2 repeat
979:C 2 read(keyboard, answer);
980:C 2 upcchar(answer);
981:C 2 until (answer='Y') or (answer='N');
982:C 2 writeln(answer);
983:C 2 yes := answer='Y';
984:C 2 end; {yes}
985:S
986:S
987:D -82 1 procedure fatal_ioreult(message: string80; iores: integer);
988:C 2 begin {fatal_ioreult}
989:C 2 writeln(#7+message);
990:C 2 getioerrmsg(message, iores);
991:C 2 writeln(' ', message);
992:C 2 escape(-1);
993:C 2 end; {fatal_ioreult}
994:S
995:S
996:D -82 1 procedure fatal_message(message: string80);
997:C 2 begin {fatal_message}
998:C 2 writeln(#7+message);
999:C 2 escape(-1);
1000:C 2 end; {fatal_message}

```



```

1001:D      1 $page$
1002:S
1003:D      1 function on_same_medium(lun1, lun2: unitnum): boolean;
1004:O      2 var
1005:D      -8 2   uep1, uep2: uep_type;
1006:C      2   begin {on_same_medium}
1007:C      2     uep1 := addr(unitable^lun1);
1008:C      2     uep2 := addr(unitable^lun2);
1009:C      2     on_same_medium := (uep1^.sc=uep2^.sc) and (uep1^.ba=uep2^.ba) and
1010:C      2       (uep1^.du=uep2^.du) and (uep1^.dv=uep2^.dv) and
1011:C      2       (uep1^.letter=uep2^.letter);
1012:C      2   end; {on_same_medium}
1013:S
1014:S
1015:D      1 function on_same_controller(uep1, uep2: uep_type): boolean;
1016:C      2   begin {on_same_controller}
1017:C      2     on_same_controller := (uep1^.sc=uep2^.sc) and (uep1^.ba=uep2^.ba);
1018:C      2   end; {on_same_controller}
1019:S
1020:S
1021:D      1 procedure enable_report_changes(uep: uep_type);
1022:C      2   begin {enable_report_changes}
1023:C      2     with uep^ do
1024:C      3     begin
1025:C      3       umediavalid := false;
1026:C      3       ureportchange := true;
1027:C      3     end; {with}
1028:C      2   end; {enable_report_changes}
1029:S
1030:S
1031:D      1 procedure disable_report_changes(uep: uep_type);
1032:C      2   begin {disable_report_changes}
1033:C      2     with uep^ do
1034:C      3     begin
1035:C      3       umediavalid := false;
1036:C      3       ureportchange := false;
1037:C      3     end; {with}
1038:C      2   end; {disable_report_changes}
1039:S
1040:S
1041:D      1 procedure user_confirmation;
1042:C      2   begin {user_confirmation}
1043:C      2     if not yes('Are you SURE you want to proceed?') then
1044:C      3     escape(-1);
1045:C      2   end; {user_confirmation}

```

```

1046:D      1 $page$
1047:S
1048:O      -82 1 procedure request_vol(prompt: string80; var lun: unitnum);
1049:S
1050:D      -82 2   const
1051:D      -82 2     nonabortive_ioresult_set = [ord(inoerror), ord(zbadblock), ord(znoblock),
1052:D      -82 2       ord(zuninitialized), ord(inodirectory)];
1053:D      -82 2   var
1054:D      -164 2     response: string80;
1055:D      -182 2     fvid: vid;
1056:D      -304 2     ftitle: fid;
1057:D      -308 2     fsegs: integer;
1058:D      -310 2     fkind: filekind;
1059:D      -311 2     scantitle_ok: boolean;
1060:D      -316 2     saved_ioresult: integer;
1061:D      -320 2     uep: uep_type;
1062:S
1063:C      2   begin {request_vol}
1064:S
1065:C      2     repeat
1066:C      3     write(prompt, cteol);
1067:C      3     readln(response);
1068:C      3     if (strlen(response)=0) or (strpos(response, #27)<>0) then escape(-1);
1069:C      3     scantitle_ok := scantitle(response, fvid, ftitle, fsegs, fkind);
1070:C      3     if scantitle_ok and (ftitle='') then
1071:C      4     begin
1072:C      4       lun := findvolume(fvid, true);
1073:C      4       saved_ioresult := ioresult;      {for decoding below}
1074:C      4       getioerrmsg(response, ord(inounit)); {for the case lun=0}
1075:C      4     end {then}
1076:C      4     else
1077:C      4     begin
1078:C      4       lun := 0;
1079:C      4       response := 'Illegal syntax: volume ID required';
1080:C      4     end; {else}
1081:C      3     if lun=0 then
1082:C      4     writeln(' ', response);
1083:C      3     until lun<>0;
1084:S
1085:C      2     uep := addr(unitable^lun);
1086:C      2     with uep^ do
1087:C      3     begin
1088:C      3       if not (letter in ['Q', 'K']) then
1089:C      4       fatal_message('Specified volume is not on a CS80 device');
1090:S
1091:C      3       write('Device: ', QdeviceName(uep, saved_ioresult));
1092:C      3       writeln(' ', i00*sc+ba:1, ' ', du:1, ' ', dv:1);
1093:C      3     end; {with uep^}
1094:S
1095:C      2     write('Logical unit #', lun:1, ' - ');
1096:C      2     if fvid[1]<>'#' then
1097:C      3     writeln(' ', fvid, ', ');
1098:C      3     else if saved_ioresult in nonabortive_ioresult_set then
1099:C      4     writeln('<no directory>');
1100:C      4     else
1101:C      4       fatal_ioresult(' ', saved_ioresult);
1102:S
1103:C      2   end; {request_vol}

```



```

1104:D      1 $page$
1105:S
1106:D      1 procedure other_volume_check(lun: unitnum);
1107:D      2
1108:D      2   var
1109:D      2     warning_issued: boolean;
1110:D      2     flun: unitnum;
1111:D      2     line: shortint;
1112:D      2     CRTline, CRTcolumn, stopline: integer;
1113:D      2     fvid: vid;
1114:D      2   const
1115:D      2     maxlines = 10;
1116:D      2     fieldwidth = 16;
1117:S
1118:C      2   begin {other_volume_check}
1119:S
1120:C      2     warning_issued := false;
1121:S
1122:C      2     for flun := 1 to maxunit do
1123:C      3       if on same_medium(flun, lun) then
1124:C      4         with unitable[flun] do
1125:C      5           begin
1126:C      6             if flun<>lun then
1127:C      7               begin
1128:C      7                 if not warning_issued then
1129:C      7                   begin
1130:C      7                     writeln('NOTICE: this will also affect:');
1131:C      7                     warning_issued := true;
1132:C      7                     line := 1;
1133:C      7                   end; {if}
1134:C      6                 fgetxy(output,CRTcolumn, CRTline);
1135:C      6                 write(' ',flun:1);
1136:C      6                 fgotoxy(output,CRTcolumn+5, CRTline);
1137:C      6                 call(dam, fvid, flun, getvolumename);
1138:C      6                 if strlen(fvid)>0
1139:C      7                   then write(fvid,':')
1140:C      7                   else write('<no dir>');
1141:C      6                 if CRTcolumn=0
1142:C      7                   then writeln      (scrolls the screen if necessary)
1143:C      7                   else fgotoxy(output,CRTcolumn, CRTline+1);
1144:C      6                 line := line+1;
1145:C      6                 if line>maxlines then
1146:C      7                   begin
1147:C      7                     fgetxy(output,CRTcolumn, stopline);
1148:C      7                     fgotoxy(output,CRTcolumn+fieldwidth, stopline-maxlines);
1149:C      7                     line := 1;
1150:C      7                   end; {if}
1151:S
1152:C      6                 end; {if flun<>lun}
1153:C      5             umediavalid := false;
1154:C      5             end; {with}
1155:S
1156:C      2     if warning_issued and (CRTcolumn<>0) then
1157:C      3       fgotoxy(output,0, stopline);
1158:S
1159:C      2   end; {other_volume_check}

```

```

1160:D      1 $page$
1161:S
1162:D      1 procedure preserving_status_do(proc: proc_type);
1163:D      2
1164:D      2   var
1165:D      2     saved_escapecode: shortint;
1166:D      2     saved_ioresult: integer;
1167:D      2   begin {preserving_status_do}
1168:D      2     saved_escapecode := escapecode;
1169:D      2     saved_ioresult := ioresult;
1170:D      2     call(proc);
1171:D      2     sysescapecode := saved_escapecode;
1172:D      2     ioresult := saved_ioresult;
1173:S
1174:S
1175:D      1 procedure writeln_proc;
1176:D      2   begin {writeln_proc}
1177:D      2     writeln;
1178:D      2   end; {writeln_proc}
1179:S
1180:S
1181:D      1 procedure under_lock_do(verbose: boolean; uep_proc: uep_proc_type;
1182:D      2     uep: uep_type; op_description: string80);
1183:D      2   begin {under_lock_do}
1184:S
1185:D      2     lockup;
1186:D      2     try
1187:D      2       if verbose then
1188:D      3         begin
1189:D      3           writeln;
1190:D      3           writeln(op_description, ' in progress');
1191:D      3         end; {if}
1192:D      2       call(uep_proc, uep);
1193:D      2       if verbose then
1194:D      3         writeln(op_description, ' completed');
1195:D      2       escape(0); {to do the lockdown}
1196:D      2     recover
1197:D      2     begin
1198:D      2       preserving_status_do(lockdown);
1199:D      2       if escapecode=0 then
1200:D      3         (do nothing)
1201:D      2       else if escapecode=-10 then
1202:D      3         begin
1203:D      3           if not verbose then
1204:D      4             preserving_status_do(writeln_proc);
1205:D      3           fatal_ioresult(op_description+' errorred:', ioresult);
1206:D      3         end
1207:D      2       else
1208:D      3         escape(escapecode);
1209:D      2       end; {recover}
1210:S
1211:C      2   end; {under_lock_do}

```



```
1212:D      1 $page$
1213:S
1214:D      1 procedure medium_copy;
1215:S
1216:D      2 var
1217:D      -4 2 source_lun, destination_lun: unitnum;
1218:D      -12 2 source_uep, tape_uep: uep_type;
1219:D      -14 2 source_is_a_tape, destination_is_a_tape: boolean;
1220:D      -16 2 source_is_a_7914, destination_is_a_7914: boolean;
1221:D      -24 2 source_block_size, destination_block_size: integer;
1222:D      -32 2 source_size, destination_size: integer;
1223:D      -33 2 verify_destination: boolean;
1224:D      -35 2 copy_attempt_completed, loop_condition: boolean;
1225:D      -38 2 pass: shortint;
1226:S
1227:D      2 procedure attempt_copy;
1228:S
1229:D      3 procedure cleanup;
1230:C      4 begin (cleanup);
1231:C      4 if destination_is_a_tape then (restore its configuration)
1232:C      5 Qconfigure(destination_uep);
1233:C      4 end; (cleanup)
1234:S
1235:C      3 begin (attempt_copy)
1236:S
1237:C      3 if source_is_a_tape and destination_is_a_7914 then (see where to restore)
1238:C      4 begin
1239:C      4 under_lock_do(silently, Qcopy_start_address, source_uep, 'Copy start address request');
1240:C      4 destination_block_address := Csa_bytes.lfb;
1241:C      4 end; (if)
1242:S
1243:C      3 if destination_is_a_tape then (ensure it's certified; update size afterward!!!)
1244:C      4 begin
1245:C      4 tapebuf.state := undefined;
1246:C      4 initialize_options_byte := 0; (certify only if currently uncertified)
1247:C      4 under_lock_do(verbosely, Qcertify, destination_uep, 'Destination tape certification');
1248:C      4 under_lock_do(silently, Qdescribe, destination_uep, 'Destination tape describe request');
1249:C      4 with describe_bytes do
1250:C      5 destination_size := (maxsvadd.lfb+1)*nbbp;
1251:C      4 end; (if)
1252:S
1253:C      3 lockup;
1254:C      3 try
1255:C      4 if destination_is_a_tape then (enable auto skip-sparing)
1256:C      5 Qtapesetup(destination_uep);
1257:S
1258:C      4 if source_is_a_7914 or destination_is_a_7914 then
1259:C      5 begin
1260:C      5 writeln;
1261:C      5 writeln('Copy parameters for 7914 save/restore -');
1262:C      5 writeln(' source starting block address: ', source_block_address:0);
1263:C      5 writeln(' destination starting block address: ', destination_block_address:0);
1264:C      5 end; (if)
1265:S
1266:C      4 under_lock_do(verbosely, Qcontrollercopy, source_uep, 'Medium copy');
1267:S
1268:C      4 if destination_is_a_tape and (destination_size>source_size) then
1269:C      5 under_lock_do(silently, Qwritefilemark, destination_uep, 'Destination tape filemark append');
1270:S
```

```
1271:C      4 escape(0); (to do the cleanup)
1272:C      4 recover
1273:C      4 begin
1274:C      4 preserving_status_do(cleanup);
1275:C      4 preserving_status_do(lockdown);
1276:C      4 if escapecode<>0 then
1277:C      5 escape(escapecode);
1278:C      4 end; (recover)
1279:S
1280:C      3 if destination_is_a_tape and verify_destination then (verify it)
1281:C      4 begin
1282:C      4 if destination_size>source_size
1283:C      5 then verify_length := source_size
1284:C      5 else verify_length := destination_size;
1285:C      4 verify_block_size := destination_block_size;
1286:C      4 under_lock_do(verbosely, Qverify, destination_uep, 'Destination tape verification');
1287:C      4 end; (if)
1288:S
1289:C      3 end; (attempt_copy)
1290:S
1291:C      2 begin (medium_copy)
1292:S
1293:C      2 writeln(clearscr);
1294:C      2 request_vol('Source medium for copy? ', source_lun);
1295:C      2 source_uep := addr(unitable^source_lun);
1296:C      2 enable_report_changes(source_uep);
1297:C      2 other_volume_check(source_lun);
1298:C      2 source_is_a_7914 := source_uep.devid=7914;
1299:C      2 with describe_bytes do
1300:C      3 begin
1301:C      3 source_is_a_tape := dt=2;
1302:C      3 source_block_size := nbbp;
1303:C      3 source_size := (maxsvadd.lfb+1)*nbbp;
1304:C      3 end; (with)
1305:S
1306:C      2 writeln;
1307:C      2 request_vol('Destination medium for copy? ', destination_lun);
1308:C      2 destination_uep := addr(unitable^destination_lun);
1309:C      2 enable_report_changes(destination_uep);
1310:C      2 if on_same_medium(source_lun, destination_lun) then
1311:C      3 fatal_message('Specified volumes are on the same medium');
1312:C      2 if not on_same_controller(source_uep, destination_uep) then
1313:C      3 fatal_message('Source and destination not on the same controller');
1314:C      2 other_volume_check(destination_lun);
1315:C      2 destination_is_a_7914 := destination_uep.devid=7914;
1316:C      2 with describe_bytes do
1317:C      3 begin
1318:C      3 destination_is_a_tape := dt=2;
1319:C      3 destination_block_size := nbbp;
1320:C      3 destination_size := (maxsvadd.lfb+1)*nbbp;
1321:C      3 end; (with)
1322:S
1323:C      2 if destination_is_a_tape and (destination_size<source_size) then
1324:C      3 if source_is_a_7914 then
1325:C      4 begin
1326:C      4 writeln;
1327:C      4 writeln('REMINDER: two long tapes are required for');
1328:C      4 writeln(' complete 7914 backup. ');
1329:C      4 end (then)
1330:C      4 else
```



```

1331:C      4      begin
1332:C      4      writeln;
1333:C      4      writeln('NOTICE: the destination tape is too small for a');
1334:C      4      writeln(' complete source backup:');
1335:C      4      end; {else}
1336:S
1337:C      2      user_confirmation;
1338:S
1339:C      2      if destination_is_a_tape
1340:C      3      then verify_destination := yes('Verify destination tape after the copy?')
1341:C      3      else verify_destination := true; {always verify discs}
1342:S
1343:C      2      pass := 1;
1344:C      2      source_block_address := 0;
1345:C      2      destination_block_address := 0;
1346:S
1347:C      2      repeat {loop for swapping tapes}
1348:S
1349:C      3      repeat {copy attempts}
1350:C      4      try
1351:C      5      attempt_copy;
1352:C      5      copy_attempt_completed := true;
1353:C      5      recover;
1354:C      5      if escapecode<>-1 then
1355:C      6      escape(escapecode)
1356:C      6      else
1357:C      6      if yes('Retry the current copy segment?') then
1358:C      7      copy_attempt_completed := false
1359:C      7      else if yes('Abort the entire medium copy sequence?') then
1360:C      8      fatal_message('Medium copy sequence aborted')
1361:C      8      else
1362:C      8      begin
1363:C      8      writeln('Medium copy sequence continued');
1364:C      8      copy_attempt_completed := true;
1365:C      8      end;
1366:C      4      until copy_attempt_completed;
1367:S
1368:C      3      if source_is_a_tape then
1369:C      4      under_lock_do(verbose, Qunload, source_uep, 'Source tape unload request');
1370:S
1371:C      3      if destination_is_a_tape then
1372:C      4      under_lock_do(verbose, Qunload, destination_uep, 'Destination tape unload request');
1373:S
1374:C      3      loop_condition := (source_is_a_7914 or destination_is_a_7914) and (pass<2);
1375:S
1376:C      3      if loop_condition then {prepare for the next pass through the loop}
1377:C      4      begin
1378:S
1379:C      4      if destination_is_a_tape then {update the source disc parms}
1380:C      5      begin
1381:C      5      source_size := source_size - destination_size;
1382:C      5      source_block_address := source_block_address + destination_size div source_block_size;
1383:C      5      end; {if}
1384:S
1385:C      4      if source_is_a_tape
1386:C      5      then tape_uep := source_uep
1387:C      5      else tape_uep := destination_uep;
1388:S
1389:C      4      writeln;
1390:C      4      writeln('Waiting for a new tape to be loaded:');

```

```

1391:C      4      writeln('Press <stop> to abort');
1392:S
1393:C      4      disable_report_changes(tape_uep);
1394:S
1395:C      4      repeat
1396:C      5      try
1397:C      6      copy_start_address(tape_uep);
1398:C      6      recover;
1399:C      6      if escapecode<>-10 then
1400:C      7      escape(escapecode)
1401:C      7      else if ioresult<>ord(znotready) then
1402:C      8      fatal_ioresult('Error while awaiting new tape load:', ioresult)
1403:C      8      until ioresult<>ord(znotready);
1404:S
1405:C      4      enable_report_changes(tape_uep);
1406:S
1407:C      4      pass := pass+1;
1408:S
1409:C      4      end; {if}
1410:S
1411:C      3      until not loop_condition;
1412:S
1413:C      2      if not destination_is_a_tape then
1414:C      3      begin
1415:C      3      verify_length := destination_size;
1416:C      3      verify_block_size := destination_block_size;
1417:C      3      under_lock_do(verbose, Qverify, destination_uep, 'Destination disc verification');
1418:C      3      end; {if}
1419:S
1420:C      2      end; {medium_copy}

```



```
1421:D      1 $page$
1422:S
1423:D      1 procedure verify;
1424:S
1425:D      2   var
1426:D      -2 2   lun: unitnum;
1427:D      -6 2   uep: uep_type;
1428:D      -7 2   auto_unload: boolean;
1429:S
1430:C      2   begin {verify}
1431:S
1432:C      2     writeln(clearscr);
1433:C      2     request_vol('Verify what medium? ', lun);
1434:C      2     uep := addr(unitable^[lun]);
1435:C      2     enable_report_changes(uep);
1436:C      2     other_volume_check(lun);
1437:S
1438:C      2     user_confirmation;
1439:S
1440:C      2     with describe_bytes do
1441:C      3       begin
1442:C      3         if dt=2
1443:C      4           then auto_unload := yes('Unload tape after verification?')
1444:C      4           else auto_unload := false;
1445:C      3         verify_length := (maxsvadd.lfb+1)*nbbp;
1446:C      3         verify_block_size := nbbp;
1447:C      3         end; {with}
1448:S
1449:C      2     under_lock_do(verbosely, Qverify, uep, 'Verification');
1450:S
1451:C      2     if auto_unload then
1452:C      3       under_lock_do(verbosely, Qunload, uep, 'Tape unload request');
1453:S
1454:C      2   end; {verify}
```

```
1455:D      1 $page$
1456:S
1457:D      1 procedure certify;
1458:S
1459:D      2   var
1460:D      -2 2   lun: unitnum;
1461:D      -6 2   uep: uep_type;
1462:D      -7 2   auto_unload: boolean;
1463:S
1464:C      2   begin {certify}
1465:S
1466:C      2     writeln(clearscr);
1467:C      2     request_vol('Certify what tape? ', lun);
1468:C      2     uep := addr(unitable^[lun]);
1469:C      2     enable_report_changes(uep);
1470:C      2     if describe_bytes.dt<>2 then
1471:C      3       fatal_message('Specified volume is not on a tape');
1472:C      2     other_volume_check(lun);
1473:S
1474:C      2     user_confirmation;
1475:S
1476:C      2     if yes('Re-certify if already certified?')
1477:C      3       then initialize_options_byte := 1 {force complete certification}
1478:C      3       else initialize_options_byte := 0; {certify only if currently uncertified}
1479:S
1480:C      2     auto_unload := yes('Unload tape after certification?');
1481:S
1482:C      2     tapebuf_state := undefined;
1483:C      2     under_lock_do(verbosely, Qcertify, uep, 'Tape certification');
1484:S
1485:C      2     if auto_unload then
1486:C      3       under_lock_do(verbosely, Qunload, uep, 'Tape unload request');
1487:S
1488:C      2   end; {certify}
```



```
1489:D      1 $page$
1490:S
1491:D      1 procedure commandlevel;
1492:S
1493:D      2   var
1494:D      -1 2   ch: char;
1495:S
1496:C      2   begin {commandlevel}
1497:S
1498:C      2       repeat
1499:S
1500:C      3         try
1501:C      4           write(homechar,
1502:C      4             'Tapebkup: Medium-copy Verify Certify-tape Quit  ?',
1503:C      4             cteol);
1504:C      4           read(keyboard, ch);
1505:C      4           upcchar(ch);
1506:C      4           writeln;
1507:C      4           case ch of
1508:C      5             'C': certify;
1509:C      5             'M': medium_copy;
1510:C      5             'Q': {quit};
1511:C      5             'V': verify;
1512:C      5             otherwise
1513:C      5               write(clearscr);
1514:C      5             end; {case}
1515:S
1516:C      4         recover
1517:C      4           if escapecode=-20 then
1518:C      5             ch := 'Q' {stop key means "quit"}
1519:C      5           else if escapecode=-1 then
1520:C      6             {do nothing; remain in command level}
1521:C      6           else
1522:C      6             escape(escapecode);
1523:S
1524:C      3       until ch='Q';
1525:S
1526:C      2   end; {commandlevel}
1527:S
1528:S
1529:S
1530:C      1 begin {tapebkup}
1531:C      1   write(clearscr);
1532:C      1   writeln('          [Version 3.0]');
1533:C      1   writeln;
1534:C      1   writeln;
1535:C      1   writeln;
1536:C      1   writeln(' Copyright 1984 Hewlett-Packard Company');
1537:C      1   writeln(' All rights are reserved. ');
1538:C      1   commandlevel;
1539:C      1 end. {tapebkup}
1540:S
1541:S
```

No errors. No warnings.

***** Nonstandard language features enabled *****

TYPES

Description

TYPES contains declarations used by the DGL graphics routines.

© Copyright Hewlett-Packard Company, 1984. This document contains proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this program except for archival purposes is prohibited without the prior written consent of Hewlett-Packard Company.

Restricted Rights Legend

Use, Duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b) (3) (B) of the Rights in Technical Data and Computer Software clause in DAR 7-104.9(a).

Pascal [Rev 3.0M 6/ 4/84] TYPES.TEXT

26-Dec-84 22:38:26 Page 2 *** File name: TYPES ***

Dump of OGL_TYPES
Importer:

DGL TYPES dump complete

```
No errors. No warnings.
***** Nonstandard language features enabled *****
```